GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

# Desarrollo de una aplicación Android de apuestas utilizando Firebase para la sincronización de datos


*Autor:*
Marina CASTELLOTE GARCÍA

Supervisor:
Sergio AGUADO GONZÁLEZ
*Tutor académico:*
Mª Ángeles LÓPEZ MALO



Fecha de lectura: 17 de julio de 2017
Curso académico 2016/2017

# Resumen

En el presente documento se detalla el desarrollo de la aplicación móvil de apuestas MyBets, para el sistema operativo Android. Este proyecto ha sido desarrollado como Trabajo Final del Grado de Ingeniería Informática en la Universitat Jaume I de Castellón, durante el curso 2016/2017. Mybets ha sido desarrollada durante la estancia en prácticas en Soluciones Cuatroochenta S.L., la empresa que ha propuesto la aplicación. La funcionalidad básica de la aplicación consiste en permitir a sus usuarios realizar apuestas sin coste y de diferentes tipos, entre las que destacan las de carácter deportivo, jugando contra otros usuarios ganando o perdiendo puntos. Para el *backend* de la aplicación se ha utilizado Firebase, tanto para la sincronización de datos en tiempo real, como para la autenticación de usuarios, el almacenamiento de archivos y el sistema de notificaciones. Para la gestión del proyecto se ha usado la metodología Scrum, incrementando la funcionalidad de la aplicación durante un total de 5 iteraciones.



# Palabras clave

Android, Firebase, Aplicación móvil, App, Grado en Ingeniería Informática, Scrum, Metodología ágil

# Keywords

Android, Firebase, Mobile application, App, Bachelor's Degree in Computer Engineering, Scrum, Agile methodology

# Índice de contenidos

# Capítulo 1

# Introducción

En este capítulo inicial se explica la necesidad que ha llevado al desarrollo de este proyecto, así como los objetivos perseguidos y la estructura general del documento.

## 1.1 Contexto y motivación del proyecto

### 1.1.1 Contexto del proyecto

El proyecto presentado en este documento se ha desarrollado como parte de la asignatura *EI1054 Prácticas Externas y Trabajo Final de Grado*, del Grado de Ingeniería Informática impartido en la Universitat Jaume I de Castellón durante el curso 2016/2017. Consiste en el desarrollo de una aplicación desarrollada para móvil, en concreto para el sistema operativo Android, utilizando Firebase como servidor *backend* donde guardar los datos.

Antes de empezar a entrar en detalles sobre el proyecto, lo cual se hará durante el capítulo 2, es necesario comentar que este ha sido propuesto por la empresa Soluciones Cuatroochenta S.L. (en adelante Cuatroochenta), en la cual la alumna ha realizado la estancia en prácticas asociada a este Trabajo Final de Grado.

Cuatroochenta tiene su sede central en Castellón, en el propio campus de la Universitat Jaume I (edificio Espaitec II), aunque también cuenta con oficinas en Panamá, Bogotá, Utrecht, Buenos Aires, Milán y Los Ángeles.

El ámbito económico de Cuatroochenta se sitúa en el campo del desarrollo de software, en concreto desarrollando aplicaciones en diferentes tipos de plataformas. Cuatroochenta trata directamente con sus clientes finales y desarrolla software personalizado para ellos, además de contar con su propia línea de productos, tales como Sefici, para la gestión de incidencias [1] y 480interactive [2], un software dedicado a crear *apps* sin conocimientos de programación. Además, la empresa colabora frecuentemente con la Universitat Jaume I dando conferencias a sus alumnos, e impulsando eventos de programación como la Hackaton de Castellón [3].

Debido a que el ámbito de desarrollo económico de esta empresa está tan relacionado con el itinerario de Ingeniería del software, la estancia en prácticas en Cuatroochenta ha aportado a la alumna una experiencia real del trabajo y ambiente laboral en una compañía especializada en desarrollo de software.

### 1.1.2 Motivación

Desde la salida de los *smartphones* al mercado, la forma de comunicación y conexión a la red ha cambiado notablemente. A medida que aumenta el número de usuarios que acceden a internet a través de sus teléfonos, son más las empresas que han observado la importancia de incrementar su visibilidad a través de *apps*.

Las aplicaciones móviles tienen un lugar permanente al lado del consumidor, el cual ya raramente se separa de su teléfono. Es por ello que son el lugar perfecto para enviar publicidad, información de la empresa, o simplemente recordar al usuario de su existencia. Los sistemas de notificaciones consiguen mantener al consumidor al día de sus productos, novedades, ofertas…, todo lo necesario para no perderlo como cliente.

Esto explica perfectamente la elección de formar a la alumna en el ámbito de desarrollo de aplicaciones en Android, ya que este es uno de los sistemas operativos más utilizados del momento en teléfonos inteligentes. Sin embargo, hay que comprender también el interés de realizar no una *app* cualquiera, sino una dedicada a apuestas. La explicación de esto es sencilla, las apuestas deportivas tienen gran popularidad entre la población, y debido a que la tecnología permite acceder a ellas sin desplazarse del hogar, las páginas de apuestas online han cosechado un gran éxito en los últimos años.

Es precisamente de esta idea combinada de donde surge la idea del proyecto; las webs de apuestas gozan de mucha popularidad, y si sumamos a esto la practicidad de poder apostar en cualquier momento, incluso mientras el usuario está disfrutando de un partido en directo, las ventajas de utilizar una aplicación móvil incrementan el atractivo del producto.

Por tanto, el auge del que gozan tanto los *smartphones* como las aplicaciones de apuestas parece indicar que un proyecto que implemente ambos puede ser considerado como una experiencia práctica de un proyecto que podría ser llevado a cabo realmente como idea de negocio.

Sin embargo, debido al carácter educativo del proyecto, que no persigue una retribución económica real, se tratará la aplicación como una alternativa gratuita a una casa real de apuestas, utilizando puntos como moneda de cambio.

## 1.2 Objetivos del proyecto

El principal objetivo del proyecto es la obtención de una aplicación nativa en Android que permita la gestión de apuestas. La función principal será permitir a los clientes de la aplicación participar en apuestas públicas, así como crear apuestas privadas para poder jugar contra sus contactos. Otras de las funciones permitidas a los usuarios serán: consultar el estado de sus apuestas, su actividad reciente y la actividad de sus amigos, ver y modificar su perfil de usuario, y añadir contactos a su lista de amigos. En el capítulo 2 se explica en detalle toda la funcionalidad del proyecto.

La aplicación se implementará desarrollando el código de la parte del cliente en lenguaje Java, y utilizando la interfaz de programación de aplicaciones (API) de Firebase para Android como servicio *backend*, el cual se utilizará para la autenticación de usuarios, el almacenamiento de datos e imágenes, y para proporcionar a la aplicación una base de datos en tiempo real.

A esto se añade el aspecto educacional del proyecto, que implica la formación de la alumna en la programación para sistemas Android, y la mejora de sus habilidades de análisis, planificación, resolución de problemas, y la capacidad de integración en un entorno de trabajo real dentro del ámbito de estudio del itinerario de ingeniería del software.

## 1.3 Estructura de la memoria

En los siguientes capítulos se explican las tareas realizadas durante la estancia en prácticas, así como las metodologías y herramientas utilizadas y los resultados finales obtenidos. La estructura de la memoria se ha establecido de la siguiente forma:

En el capítulo 2 se describe el proyecto, incluyendo la explicación de las tecnologías utilizadas.

En el capítulo 3 se pretende introducir el método seguido a la hora de planificar el proyecto, detallando en sus diferentes secciones las distintas fases llevadas a cabo durante el seguimiento del proyecto, añadiendo además información relevante como la estimación de costes.

En el capítulo 4 se explica el análisis del sistema. También se explica el diseño de la aplicación en sus diferentes partes, incluyendo tanto el diseño de la base de datos, como de la arquitectura y de la interfaz de usuario.

En el capítulo 5 se detalla la implementación y las pruebas de verificación realizadas.

Finalmente, las conclusiones obtenidas de la realización de este proyecto quedan resumidas en el capítulo 6.

# Capítulo 2

# Descripción del proyecto

## 2.1 Descripción y funciones

Como se ha mencionado en el capítulo 1, este proyecto consiste en el desarrollo de una aplicación de apuestas, principalmente deportivas, desarrollada para el sistema operativo Android, abarcando todo el apartado de desarrollo de la aplicación y del módulo de pruebas.

Para poder llevar a cabo el desarrollo, la empresa ha proporcionado a la alumna un análisis de requisitos inicial, el cual comprende exclusivamente el diseño de las pantallas de la interfaz de usuario. Esto se muestra en más detalle en el apartado 3.1 de este documento, donde se explica la metodología de trabajo llevada a cabo. Por su parte, el diseño de la interfaz puede ser consultado en el apartado 4.5. El nombre de la aplicación, decidido por la empresa, es MyBets.

Es importante destacar que el proyecto no comprende el lanzamiento real de la aplicación en ninguna plataforma de acceso público, puesto que ha sido ideado únicamente como una tarea completa de formación en programación para Android, sin expectativas de un beneficio económico real.

MyBets es una *app* donde los usuarios pueden realizar apuestas, y aunque se centra sobre todo en el ámbito del deporte, también se incluyen opciones de apuestas en otros sectores como por ejemplo: música, televisión, política, sociedad, etc., aunque durante el desarrollo de aplicación no se ha profundizado en ninguna de estas categorías tanto como en la de deporte, ya que solo existen para ampliar el público objetivo y el atractivo como juego de entretenimiento.

Además, las apuestas realizadas a través de la aplicación no serán nunca hechas con dinero real, sino con un sistema de puntos, estableciendo cada apuesta como un juego entre los usuarios.

La aplicación cuenta con cuatro secciones principales:

- Sección “Apuestas”: Un apartado con todos los juegos que están en marcha actualmente en la aplicación. Los usuarios pueden seleccionar la apuesta deseada y participar en ella. Estas apuestas pueden ser públicas, participando contra todos los jugadores, o privadas, de forma

que el usuario puede seleccionar de su lista de amigos aquellos contra los que quiere jugar e invitarlos a unirse a su juego.

- Sección “Mis Apuestas”: En esta sección el usuario puede consultar todas las apuestas que tiene en marcha, tanto las que ya han sido jugadas por él como aquellas a las que ha sido invitado por otro jugador (a las que puede unirse o rechazar si lo desea). Es aquí donde el usuario podrá consultar la clasificación de resultados una vez las apuestas hayan finalizado.

- Sección “Muro”: Aquí el usuario recibirá mensajes detallando la actividad de sus amigos.

- Sección “Amigos”: Este apartado se usa para que el usuario pueda añadir amigos, para posteriormente poder invitarlos a jugar apuestas privadas.

También hay secciones para poder consultar los diferentes juegos filtrados según las diferentes categorías a las que pertenecen, un perfil de usuario donde poder cambiar los datos o la imagen de usuario, y un apartado de configuración donde elegir si se quiere o no recibir notificaciones en el teléfono.

La aplicación también permite registrarse tanto con un email y contraseña como a través de Facebook, así como vincular ambos tipos de cuenta si se desea buscar amigos que también se tengan agregados en Facebook.

Para entender el funcionamiento central de la aplicación, se muestra a continuación un ejemplo de cómo un usuario cualquiera realizaría una apuesta:

1. Un usuario busca en la aplicación la apuesta en la que le gustaría participar:

   El usuario selecciona la competición (por ejemplo la Liga Española de fútbol). Las competiciones tienen asignada una modalidad de juego de entre tres posibles: a un ganador (se selecciona el ganador de entre una lista), 1x2 (se selecciona si el resultado de un enfrentamiento lo ganará el contrincante 1, el contrincante 2, o acabará en empate, lo cual se simboliza con una “x”), o por puntos (se seleccionan los puntos exactos que anotará cada contrincante en un enfrentamiento).
2. El usuario elige si quiere participar en la apuesta pública o privada.
   a. Si el usuario ha seleccionado la apuesta pública, se pasa al punto 3.
   b. Si el usuario ha seleccionado la apuesta privada, pasa a seleccionar los amigos contra los que quiere jugar, a los que se les mandará una invitación al juego. Después se pasa al punto 3.
3. El usuario realiza la apuesta.
4. Cuando los resultados de las apuestas están listos, los usuarios pueden consultarlas y si han ganado puntos, se añaden a su cuenta.

MyBets es por tanto una aplicación de ocio y de tipo social, creada con el objetivo de entretener al usuario.

Cabe también explicar que no es esta la primera vez que Cuatroochenta propone este proyecto como idea para un Trabajo Final de Grado, sino que ya fue desarrollado anteriormente por otros tres alumnos durante el curso 2015/2016. Sin embargo, las diferencias entre este proyecto y los anteriores son notables y se detallan a continuación:

- *"Realización de los servicios web para una aplicación móvil de apuestas deportivas"*:

  Tal y como se puede consultar en el resumen publicado en el repositorio de la UJI [4], este proyecto abordó el desarrollo de un *backend* mediante la programación de servicios web REST (Representational State Transfer), los cuales después utilizaron los otros dos proyectos en sus aplicaciones para comunicarse con el servidor. Además, también se encargó de la realización de un portal web para permitir a los administradores poblar la base de datos. Sin duda este es el proyecto que menos parecido tiene con el actual, ya que en este último no se han desarrollado servicios REST, ni se ha creado una plataforma para poblar la base de datos, ya que la comunicación con el *backend* se ha realizado completamente con Firebase, y la base de datos se alimenta con información directamente insertada mediante la interfaz que Firebase proporciona en su web.

- *"Desarrollo de aplicación iOS utilizando webservices REST en formato JSON"* [5]:

  Este proyecto desarrolló el código cliente de la aplicación MyBets para sistemas operativos iOS, utilizando el lenguaje de programación Swift, y para la comunicación con el servidor utilizó los servicios REST desarrollados en el proyecto ya comentado. En cambio, el presente proyecto ha desarrollado el código cliente en Android, utilizando Java, y como ya se ha explicado, se ha usado Firebase como servicio *backend*, por lo que el desarrollo ha sido muy distinto.

- *"Desarrollo de aplicación Android para proyecto de apuestas deportivas"* [6]:

  Este es el proyecto que más se parece al explicado en este documento, puesto que en él también se desarrolló el código cliente de MyBets para sistemas operativos Android. Sin embargo, la principal diferencia aquí, radica en la implementación del *backend*. Mientras que la aplicación desarrollada durante el curso 2015/2016 ha utilizado los servicios REST proporcionados por otro proyecto, en el proyecto actual se ha utilizado Firebase para proporcionar un *backend* a la aplicación. Se ha utilizado tanto para la lectura y escritura en la base de datos, como el almacenamiento de archivos, el sistema de notificaciones y la autenticación de los usuarios, que se ha realizado usando Firebase Authentication [7]. Otra diferencia importante es que Firebase utiliza una base de datos noSQL, por lo que la estructura de la base de datos realizada es totalmente diferente a la del otro proyecto. También ha sido necesario realizar un análisis completo de requisitos de datos para poder estructurar la base de datos, lo cual no fue necesario en el anterior proyecto debido a que este utilizó los servicios REST que se le proporcionaron.

Como puede verse, dos de los proyectos realizaron la parte del cliente de la aplicación, utilizando los servicios REST desarrollados en el tercer proyecto para comunicarse con el servidor. Este es el ejemplo perfecto para resaltar la utilidad de Firebase en aplicaciones móvil donde no se dispone de mucho tiempo para desarrollar el *backend* de la aplicación. En el caso mencionado se necesitó un proyecto entero de prácticas para realizar los servicios REST, mientras que Firebase

proporciona un *backend* como servicio, de forma que a través de su API se pueden guardar y sincronizar datos en tiempo real en los servidores de Firebase sin que el desarrollador necesite programar una gran cantidad de código en la parte del servidor. Además, en este proyecto se ha utilizado una base de datos noSQL con una estructura de árbol JSON, al contrario que en los del año anterior, en los cuales se usó una base de datos SQL. Otro detalle importante es la seguridad del proyecto. Al usar Firebase, los permisos para realizar operaciones con la base de datos se controlan completamente desde la consola de Firebase, que internamente administra la seguridad utilizando SSL (Secure Sockets Layer) con claves de 2048 bits para sus certificados [8], mientras que al usar servicios web con peticiones HTTP (Hypertext Transfer Protocol), existe un riesgo de seguridad si no se controlan adecuadamente.

El punto de partida del presente proyecto ha sido exclusivamente el diseño de las pantallas de la aplicación, proporcionadas por la empresa. No se ha utilizado información de los tres proyectos explicados anteriormente como referencia en ningún caso.

Qué es Firebase y dónde encaja en este proyecto se explicará en más detalle en siguientes apartados de este capítulo.

## 2.2 Alcance

Respecto al alcance de la aplicación dentro de la empresa, esta sería considerada como un producto del departamento de desarrollo Android, que se encarga de la programación (en este caso esto lo ha desarrollado completamente la alumna), trabajando conjuntamente con los diseñadores, que son los encargados de determinar todo el aspecto visual de la aplicación.

En cuanto al alcance funcional, como resumen de las funciones de la aplicación el sistema debe permitir a los usuarios de esta:

- Registrarse e iniciar sesión en la aplicación.
- Participar en juegos públicos sin invitación.
- Crear apuestas privadas invitando a sus contactos a participar a un juego.
- Conectarse con su Facebook para poder encontrar a sus contactos más fácilmente.
- Buscar y consultar las diferentes categorías de deportes y competiciones.
- Ganar y perder puntos según el resultado de sus apuestas.
- Consultar los juegos más populares.
- Aceptar o denegar invitaciones a participar en juegos privados.
- Consultar el estado de sus apuestas, así como consultar en qué juegos tiene pendiente una apuesta y el resultado de las que ya han finalizado.
- Ver su muro de actividad, donde se le informa de los puntos ganados por sus contactos.

- Ver su perfil, donde pueden consultar su saldo de puntos y sus últimos puntos ganados o perdidos.
- Modificar los datos de su perfil.
- Decidir si quiere o no recibir notificaciones.

## 2.3 Tecnologías utilizadas

### 2.3.1 Realizando una aplicación nativa en Android: Motivos y herramientas

A la hora de desarrollar una aplicación móvil se puede elegir entre realizarla en el lenguaje nativo de la plataforma, en este caso Java, o desarrollar una aplicación híbrida, o una aplicación web. Aunque la elección depende de diversos factores, como por ejemplo si se va a reutilizar el código o el coste que se quiera destinar al desarrollo, es cierto que las aplicaciones nativas presentan una serie de ventajas:

- En primer lugar, las guías de diseño: En el caso de Android, se dispone de Material Design [9], una normativa de diseño desarrollada por Google que nos da una serie de pautas para que las aplicaciones tengan un aspecto visual que utilice principios de buen diseño, y que también dé al usuario una experiencia unificada en las distintas plataformas y dispositivos de tamaños diferentes. La aplicación MyBets ha seguido estos principios de forma consistente en el diseño de su interfaz, de forma que el aspecto es moderno y consistente con otras aplicaciones en las que el usuario ya tenga experiencia.
- Otra ventaja es el rendimiento de la aplicación. Una *app* nativa se optimiza específicamente para un sistema operativo en concreto, se adapta mejor al hardware del dispositivo, y por ello proporcionan una mejor experiencia al usuario.
- Una aplicación nativa presenta mayor seguridad que por ejemplo una aplicación web, ya que esta depende de la seguridad del navegador, y es más fácil de desarrollar que una aplicación híbrida.

En el caso de MyBets, se ha utilizado la herramienta oficial preparada por Google específicamente para la realización de aplicaciones nativas para Android, el entorno de desarrollo integrado (IDE) Android Studio. Este IDE incorpora herramientas especialmente preparadas para editar, depurar y probar código de desarrollo de aplicaciones Android. Entre sus características destacan [10]:

- Instant Run: Función que permite que los cambios en el código se apliquen en ejecución, de forma que el desarrollo es más rápido, sin necesidad de compilar de nuevo la aplicación.
- Editor inteligente: A medida que se escribe código, Android Studio muestra en una lista desplegable las funciones disponibles a utilizar, y proporciona documentación.

- Emulador: El uso del emulador de Android permite probar la aplicación en diferentes tamaños de dispositivos, sin necesitar tener uno real. También se pueden simular algunas de las funciones del hardware, como las funciones táctiles.

Las tres características aquí descritas han sido cruciales a la hora de agilizar el desarrollo de la aplicación, especialmente el uso del editor inteligente para disponer de la documentación necesaria a medida que se está programando.

### 2.3.2 Firebase

Gran parte de este proyecto ha consistido en la exploración y uso de varias de las funcionalidades que Firebase proporciona para aplicaciones móviles. De ahí la importancia en este proyecto de explicar qué es Firebase y cómo puede ser útil en una aplicación de estas características.

Firebase es una plataforma desarrollada por Google que facilita el desarrollo de *apps*, proporcionando un servidor *backend* para las aplicaciones. Además, el mismo *backend* puede ser utilizado de forma común en diversas plataformas: Android, IOS y web.

Firebase proporciona una solución eficaz frente no solo a problemas de desarrollo, sino también de escalabilidad a medida que la base de usuarios de la aplicación crece, ya que los servidores son proporcionados por Google. Entre sus funcionalidades se encuentra un servicio de autenticación, base de datos en tiempo real, almacenamiento de archivos, solución de errores, funciones *backend*, testeo, y medida de estadísticas recogidas de los usuarios. Aunque al realizar MyBets no se ha hecho uso de todas estas funciones, sí se ha usado una gran parte. A continuación se explica el papel que ha tenido cada una de ellas en la aplicación [11]:

- Base de datos en tiempo real: Firebase proporciona una base de datos noSQL que almacena datos y los sincroniza en tiempo real. Esto ha sido utilizado para almacenar todos los datos de las apuestas realizadas, los resultados, datos de los usuarios como los puntos de los que disponen, y en general, cualquier dato de la aplicación que necesite ser guardado.

- Autenticación: Firebase proporciona un método de registro e inicio de sesión que no solo incluye autenticación a través de correo, sino que también permite la autenticación a través de proveedores externos como Facebook, Twitter, Github y Google. En el caso de MyBets se ha utilizado tanto el registro por correo como con el proveedor de Facebook, utilizando una API destinada para ello.

- Almacenamiento de archivos: Esta característica se ha utilizado para almacenar ciertas imágenes que la aplicación utiliza, así como las fotos de perfil del usuario. Aunque algunas de las imágenes se podrían haber guardado de forma local, se ha decidido hacerlo de esta forma para garantizar que se puedan añadir nuevos juegos y categorías a la aplicación sin necesidad de que los usuarios la actualicen, simplemente utilizando la base de datos y el almacenamiento.

- Funciones *backend*: Firebase permite ejecutar código Javascript en el servidor. Esto ha sido útil en Mybets para código que no puede ser ejecutado en la parte del cliente, como por

ejemplo notificaciones de tipo *push* que han de llegar a una serie de dispositivos cuando se produce un cambio en la base de datos.

Se discutirá en más detalle la repercusión que ha tenido el uso de esta tecnología en el capítulo 5, donde se explican detalles sobre la implementación.

# Capítulo 3

# Planificación del proyecto

## 3.1 Metodología

### 3.1.1 El proceso de desarrollo

Para entender la metodología que se ha seguido, primero hay que entender la forma de trabajar de la empresa en la que se han realizado las prácticas, Cuatroochenta.

En una situación real, cuando un cliente se pone en contacto con Cuatroochenta para contratar el desarrollo de una *app* personalizada, el primer paso es la redacción de un documento corto con unos requisitos básicos. Esto lo hace el departamento de gestión junto con el cliente. Una vez ambas partes se ponen de acuerdo en esos requisitos iniciales, el equipo de diseño realiza la parte visual de la aplicación en las plataformas correspondientes, ya sea web, móvil, etc. Hasta que el diseño no está acabado, no se pasa el proyecto al grupo de desarrollo. A partir del diseño, Cuatroochenta puede estimar la duración del proyecto y los costes de la aplicación, y es cuando puede dar un presupuesto al cliente. Esta es la metodología que la empresa, a través de su experiencia, ha encontrado que se adecúa mejor a su modelo de trabajo.

En el caso de MyBets, Cuatroochenta incorpora al estudiante principalmente en la etapa de desarrollo del proyecto. Es a partir de aquí donde empieza el trabajo de la alumna, la cual individualmente se encarga de realizar el análisis de requisitos, el diseño de la aplicación y modelo de datos, y el desarrollo de la aplicación, siempre tratando de respetar los *mockups* de la interfaz de usuario dados por la empresa, junto con la opinión del supervisor. La siguiente etapa es la realización de pruebas de la aplicación. En esta ocasión será la propia alumna quien se encargue también de este apartado.

Por tanto, debido a que uno de los objetivos es la integración de la alumna en la empresa, la metodología que se ha seguido en el desarrollo de MyBets ha sido parecida a la que usa Cuatroochenta, ágil en su mayor parte. Tanto el análisis y el diseño de la aplicación como las pruebas, se han realizado periódicamente a medida que se avanzaba la programación del proyecto, y han estado abiertos a los cambios necesarios. En concreto se ha realizado la metodología Scrum.

Cuando se habla de Scrum [12], es importante comprender que su principal ventaja frente a otros acercamientos a la gestión de proyectos más tradicionales, es la división que realiza del proceso de trabajo en partes más pequeñas. Con una metodología tradicional, como por ejemplo la de cascada, primero se realiza la tarea de planificación completa, después se pasa a la etapa de desarrollo, a continuación se prueba y revisa el producto, y finalmente se entrega. La diferencia con Scrum es que este proceso que normalmente dura meses o en ocasiones incluso años, se divide en partes pequeñas en ciclos de 1 a 4 semanas. En el caso de Mybets se han realizado 5 sprints de dos semanas, menos el quinto que ha durado solamente una semana. Al dividir de esta forma el proceso, se puede asegurar que el producto siempre se corresponda con lo que el mercado necesita, y se aumenta la probabilidad de acabarlo con éxito.

Al ser el supervisor el encargado de proporcionar el diseño de la interfaz a la alumna, es este quien representa al cliente final de la aplicación, proporcionando una retroalimentación periódica a través de pequeñas reuniones al ir completándose las funcionalidades de la aplicación. Estas reuniones se producen al final de cada sprint.

Por otro lado, la alumna asume el resto de roles de Scrum, siendo estos el de Scrum Manager (encargándose de organizar las tareas a realizar en cada sprint) y como programadora y *tester* de la aplicación.

Scrum también implementa reuniones periódicas en las que se mencionan los avances conseguidos en el proyecto, así como reuniones de planificación y revisión de cada sprint. En este caso las reuniones pertinentes de seguimiento se han realizado con el supervisor del proyecto.

Para poder hacer el seguimiento de las tareas, se ha creado la pila del producto o *backlog* de la aplicación, el cual incluye todas las funcionalidades que se han de incluir en la *app*, expresadas en forma de historias de usuario, y se ha puntuado cada una de ellas utilizando el método de los puntos de historia [13].

Durante la planificación de cada sprint se han decidido cuáles de las tareas del *backlog* pasan al siguiente sprint para ser implementadas. Esto se ha repetido en cada uno de los 5 ciclos incrementales.

### 3.1.2 Herramientas utilizadas

Además del IDE Android Studio, el cual ya ha sido mencionado en el apartado 2.3 de este documento, se han utilizado otras herramientas para la gestión del proyecto:

Para tener claras las tareas a realizar y el nivel de avance del proyecto se ha hecho uso del software Jira, a través de su plataforma web, la cual es utilizada por Cuatroochenta como herramienta de gestión de proyectos de software [14]. Jira es una herramienta que los equipos ágiles utilizan a menudo en sus proyectos, ya que proporciona tanto seguimiento de errores e incidencias como tableros para planificar los sprints, y control de cada una de las tareas a realizar. Inicialmente, la plataforma Jira fue creada para el desarrollo de software, lo cual la convierte en una herramienta que se adapta adecuadamente a este propósito. En esta plataforma se han ido registrando todas las tareas a realizar durante el proyecto, así como el tiempo dedicado a trabajar en cada una de ellas,

incluyendo también las incidencias de errores ocurridos durante la implementación. Esta lista de tareas será vista en más detalle en el apartado de seguimiento del proyecto de este mismo capítulo.

Otra herramienta que se ha utilizado es un repositorio en Bitbucket [15], una plataforma para el control de versiones de código. El control de versiones es especialmente interesante al desarrollar conjuntamente con un equipo de desarrolladores, sin embargo, al ser esta aplicación un proyecto individual, se ha utilizado sobre todo para controlar y comparar las diferentes versiones del proyecto a nivel personal. Ha sido especialmente útil para ver los cambios realizados cuando han surgido errores en la ejecución de la aplicación que no se daban en versiones anteriores del código.

También cabe mencionar la utilización del software Adobe Illustrator [16], una aplicación de gráficos vectoriales que se utiliza para crear logotipos, iconos, tipografías, *mockups*… Este programa se ha utilizado para exportar los recursos necesarios para completar el aspecto visual de la aplicación. Recursos que como ya se ha mencionado han sido proporcionados por Cuatroochenta.

Por último, también se ha utilizado la plataforma Udemy [17] para realizar un curso de aprendizaje de Android antes de comenzar el desarrollo de la aplicación.

## 3.2 Planificación del proyecto

Esta sección detalla la lista de actividades y tareas que se han desarrollado durante el proyecto a nivel global, no solo a nivel de desarrollo de la aplicación.

En primer lugar, en la Figura 1 se puede observar la lista de tareas del proyecto. Estas han sido clasificadas en actividades: Inicio, planificación, análisis y diseño, desarrollo, cierre y documentación del trabajo final de grado.

En las Figuras 2 y 3 se adjunta la estructura de descomposición del trabajo (EDT), donde se aprecia la planificación inicial y las dependencias entre tareas. Como puede verse, las únicas tareas que se desarrollan en paralelo son aquellas relacionadas con la documentación, ya que se realizan a lo largo de toda la programación de la aplicación. Sin embargo las tareas de implementación nunca se solaparán entre sí, debido a que siempre serán desarrolladas por la misma persona.

El total de horas dedicado a la asignatura *EI1054 Prácticas Externas y Trabajo Final de Grado* es de 450 (18 créditos), divididas como sigue:

- 300 horas presenciales de prácticas en la empresa
- 150 horas para la elaboración de la memoria y documentación asociada al Trabajo Final de Grado (TFG)

Si se observa la lista de tareas de las Figuras 2 y 3, se debe tener en cuenta que las tareas comprendidas en la elaboración de la documentación (las cuales serán realizadas fuera del horario de prácticas en la empresa), son:

- Definición del proyecto
- Identificación del alcance y la funcionalidad
- Realización de la propuesta técnica
- Elaboración de informes quincenales
- Elaboración memoria
- Elaboración presentación
- Presentación oral TFG
- Revisión de la memoria

Sumadas dan un total de 150 horas, lo cual deja otras 300 horas de prácticas de empresa.

Por último, en el anexo A se ha adjuntado el diagrama temporal del proyecto. La estancia en prácticas acaba con la entrega del proyecto, mientras que el desarrollo del TFG continúa. Las fechas de presentación y revisión de memoria son orientativas, ya que aún no han sido determinadas.

## 3.3 Pila del producto de la aplicación

La pila del producto recoge la lista de funcionalidades que se desea que la aplicación permita realizar a sus usuarios. Esta ha sido elaborada al principio de la planificación, dando a cada una de las funcionalidades la forma de historias de usuario. Se ha utilizado el método de estimación por puntos de historia para estimar la complejidad de cada funcionalidad [13], y se ha seguido la sucesión de Fibonacci para establecer los diferentes niveles de complejidad: 1 - 2 - 3 - 5 - 8.

Este método ha sido útil a la hora de planificar los diferentes sprints, de forma que el número de puntos de historia fuera equilibrado para cada uno de ellos. En la Tabla 1 se puede consultar la pila de producto inicial de la aplicación.

El total de puntos de historia iniciales es de 50. Como se ha dicho anteriormente se ha decidido realizar 4 sprints de 2 semanas y un último sprint de 1 semana, así que los puntos a repartir entre los diferentes sprints son aproximadamente 11 puntos de historia en cada uno de los primeros 4 sprints, y 6 puntos de historia en el último sprint, para realizar un desarrollo equilibrado. Sin embargo, como se explica en el seguimiento del proyecto (apartado 3.6 de este documento), la estimación inicial ha cambiado en algunas tareas durante el curso del desarrollo de la aplicación.

Inicio

- Definición del proyecto
- Definir la metodología y los estándares de trabajo
- Identificación del alcance y las funcionalidades (pila del producto)
- Periodo de formación en desarrollo de aplicaciones Android

Planificación

- Entrega de la propuesta técnica
- Planificar tareas y estimar de puntos de historia

Análisis y diseño

- Análisis del modelo de datos
- Identificación de los requisitos de datos
- Validación del análisis con el supervisor

Desarrollo por sprints

- Planificación sprint
- Programación sprint
- Revisión y pruebas

Cierre

- Entrega proyecto

Documentación TFG

- Redacción de la propuesta técnica
- Elaboración de informes quincenales
- Elaboración de la memoria
- Elaboración de la presentación
- Presentación oral TFG
- Revisión de la memoria

Figura 1: Lista de tareas del trabajo final de grado

| | Nombre de tarea | Duración | Comienzo | Fin | Predecesora |
|---|---|---|---|---|---|
| 1 | **Proyecto MyBets** | **450 horas** | **lun 27/02/17** | **vie 28/07/17** | |
| 2 | **Inicio** | **72 horas** | **lun 27/02/17** | **jue 16/03/17** | |
| 3 | Definición del proyecto | 1 hora | lun 27/02/17 | lun 27/02/17 | |
| 4 | Definición de metodología y estándares de trabajo | 2 horas | lun 27/02/17 | lun 27/02/17 | 3 |
| 5 | Identificación alcance y funcionalidad | 3 horas | lun 27/02/17 | mar 28/02/17 | 4 |
| 6 | Periodo formación en desarrollo aplicaciones Android | 66 horas | mar 28/02/17 | jue 16/03/17 | 5 |
| 7 | **Planificación** | **2 horas** | **jue 16/03/17** | **jue 16/03/17** | **5** |
| 8 | Planificar tareas y estimar puntos de historia | 2 horas | jue 16/03/17 | jue 16/03/17 | 5 |
| 9 | Entrega propuesta técnica | 0 horas | jue 16/03/17 | jue 16/03/17 | 8 |
| 10 | **Análisis y diseño** | **5 horas** | **vie 17/03/17** | **vie 17/03/17** | |
| 11 | Análisis del modelo de datos | 2 horas | vie 17/03/17 | vie 17/03/17 | 8 |
| 12 | Identificación requisitos de datos | 2 horas | vie 17/03/17 | vie 17/03/17 | 11 |
| 13 | Validación del análisis | 1 hora | vie 17/03/17 | vie 17/03/17 | 12 |
| 14 | **Desarrollo** | **225 horas** | **mar 21/03/17** | **vie 02/06/17** | |
| 15 | Planificar sprint 1 | 2 horas | mar 21/03/17 | mar 21/03/17 | 13 |
| 16 | Programar sprint 1 | 44 horas | mié 22/03/17 | lun 03/04/17 | 15 |
| 17 | Revisar sprint 1 | 4 horas | mar 04/04/17 | mar 04/04/17 | 16 |
| 18 | Planificar sprint 2 | 2 horas | mié 05/04/17 | mié 05/04/17 | 17 |
| 19 | Programar sprint 2 | 44 horas | jue 06/04/17 | mié 26/04/17 | 18 |
| 20 | Revisar sprint 2 | 4 horas | jue 27/04/17 | jue 27/04/17 | 19 |
| 21 | Planificar sprint 3 | 2 horas | vie 28/04/17 | vie 28/04/17 | 20 |
| 22 | Programar sprint 3 | 44 horas | mar 02/05/17 | jue 11/05/17 | 21 |
| 23 | Revisar sprint 3 | 4 horas | vie 12/05/17 | vie 12/05/17 | 22 |
| 24 | Planificar sprint 4 | 2 horas | lun 15/05/17 | lun 15/05/17 | 23 |
| 25 | Programar sprint 4 | 44 horas | lun 15/05/17 | vie 26/05/17 | 24 |
| 26 | Revisar sprint 4 | 4 horas | vie 26/05/17 | vie 26/05/17 | 25 |
| 27 | Planificar sprint 5 | 2 horas | lun 29/05/17 | lun 29/05/17 | 26 |
| 28 | Programar sprint 5 | 19 horas | mar 30/05/17 | jue 01/06/17 | 27 |
| 29 | Revisar sprint 5 | 4 horas | vie 02/06/17 | vie 02/06/17 | 28 |

Figura 2: Estructura de descomposición del trabajo (I)

| | | | | | |
|---|---|---|---|---|---|
| 30 | **Cierre** | **0 horas** | **vie 30/06/17** | **vie 30/06/17** | |
| 31 | Entrega proyecto | 0 horas | vie 30/06/17 | vie 30/06/17 | 29 |
| 32 | **Documentación TFG** | **146 horas** | **lun 27/02/17** | **vie 28/07/17** | |
| 33 | Redacción de la propuesta técnica | 15 horas | lun 27/02/17 | jue 16/03/17 | 4 |
| 34 | Elaboración informes quincenales 1 | 1 hora | mié 12/04/17 | mié 12/04/17 | 33 |
| 35 | Elaboración informes quincenales 2 | 1 hora | jue 27/04/17 | jue 27/04/17 | 34 |
| 36 | Elaboración informes quincenales 3 | 1 hora | vie 12/05/17 | vie 12/05/17 | 35 |
| 37 | Elaboración informes quincenales 4 | 1 hora | mar 27/06/17 | mar 27/06/17 | 36 |
| 38 | Elaboración de la memoria | 95 horas | lun 03/04/17 | vie 30/06/17 | 37 |
| 39 | Elaboración de la presentación del TFG | 26 horas | lun 03/07/17 | mié 26/07/17 | 38 |
| 40 | Presentación oral TFG | 1 hora | jue 27/07/17 | jue 27/07/17 | 39 |
| 41 | Revisión de la memoria | 5 horas | vie 28/07/17 | vie 28/07/17 | 40 |
| 42 | | | | | |

Figura 3: Estructura de descomposición del trabajo (II)

## 3.4 Planificación de los sprints

Como se ha explicado en la sección de metodología, una vez finalizado el periodo de aprendizaje inicial de desarrollo de aplicaciones Android, se han realizado los sprints, divididos como se puede observar en la Tabla 2. Los sprints se extienden por tanto en periodos de 2 semanas (10 días laborables), menos el último que dura 1 semana (5 días laborables). La alumna ha realizado 5 horas de prácticas diarias, por lo que se puede estimar que los primeros 4 sprints tienen una duración de 50 horas los primeros 4, y 25 horas el último. Sin embargo, en estas horas se incluyen también las planificaciones y revisiones de los sprints que se realizan al inicio y final de cada uno de ellos. Además hay que tener en cuenta que no hay una equivalencia directa entre el número de puntos de historia asignados a una historia de usuario y las horas que se necesitan para su implementación, aunque lógicamente, a mayor complejidad es probable que se necesite más tiempo para realizar una tarea.

| Id | Descripción historia de usuario | Nº de puntos de historia asociados |
|---|---|---|
| HU01 | Como usuario necesito poder registrarme en la aplicación con mi nombre y contraseña o con mi cuenta de Facebook para poder acceder posteriormente a mi cuenta. | 3 |
| HU02 | Como usuario necesito poder vincular una cuenta registrada con correo a mi cuenta de Facebook para poder acceder a mis contactos de Facebook. | 2 |
| HU03 | Como usuario necesito poder iniciar sesión en la aplicación con mi cuenta y contraseña o con mi cuenta de Facebook. | 2 |
| HU04 | Como usuario necesito poder ver mi perfil de usuario y ver mi actividad. | 3 |
| HU05 | Como usuario necesito poder modificar mi perfil de usuario. | 2 |
| HU06 | Como usuario necesito poder decidir si quiero recibir notificaciones de la aplicación en mi dispositivo. | 1 |
| HU07 | Como usuario necesito poder listar las apuestas disponibles en la aplicación clasificadas según diferentes criterios: apuestas publicadas en el día actual, apuestas populares, apuestas clasificadas según las diferentes categorías disponibles. | 8 |
| HU08 | Como usuario necesito poder ver la información de las apuestas. | 3 |
| HU09 | Como usuario necesito poder unirme a una apuesta pública de la aplicación. | 5 |
| HU10 | Como usuario necesito poder realizar una apuesta privada. | 5 |
| HU11 | Como usuario necesito poder invitar a mis amigos a jugar en mis apuestas privadas. | 2 |

| | | |
|---|---|---|
| HU12 | Como usuario necesito poder consultar mis apuestas en marcha y ver su estado actual. | 3 |
| HU13 | Como usuario necesito poder jugar o rechazar una apuesta a la que me han invitado. | 1 |
| HU14 | Como usuario necesito poder consultar la clasificación de los jugadores en una apuesta de la que ya se conocen los resultados. | 2 |
| HU15 | Como usuario necesito poder consultar las apuestas de mis contrincantes así como las mías, y ver mis resultados comparados con los resultados correctos. | 2 |
| HU16 | Como usuario necesito poder consultar la lista de usuarios de la aplicación y agregarlos a mi lista de amigos. | 2 |
| HU17 | Como usuario necesito poder ver la actividad de mis amigos. | 1 |
| HU18 | Como usuario necesito poder recibir notificaciones en mi dispositivo. | 3 |

Tabla 1: Pila del producto inicial

| **Sprint** | **Fecha de inicio del sprint** | **Fecha de fin del sprint** |
|---|---|---|
| 1 | 21 de marzo de 2017 | 4 de abril de 2017 |
| 2 | 5 de abril de 2017 | 27 de abril de 2017 |
| 3 | 28 de abril de 2017 | 12 de mayo de 2017 |
| 4 | 15 de mayo de 2017 | 26 de mayo de 2017 |
| 5 | 29 de mayo de 2017 | 2 de junio de 2017 |

Tabla 2: Periodos de duración de los sprints

## 3.5 Estimación de recursos y costes del proyecto

A continuación se detallan los gastos estimados de desarrollo, comprendiendo tanto recursos humanos como materiales y herramientas de software utilizadas.

Para el cálculo se ha tenido en cuenta el salario bruto estándar de un ingeniero junior de 20.000 euros anuales. Añadiendo a esto el IRPF, Seguridad Social y otras contribuciones que la empresa aporta debido al trabajador, y teniendo en cuenta que las horas laborables efectivas anuales de un trabajador son 1780, se ha extraído el coste del desarrollador, como se ve en la Tabla 3.

| **Salario bruto** | **Coste anual** | **Coste / horas laborables efectivas anuales** | **Total** |
|---|---|---|---|
| 20.000 € | 32.180 € | 18,08€/hora | 18,08€ x 300horas = 5.424€ |

Tabla 3: Costes asociados a recursos humanos

Para el cálculo de la amortización del ordenador y otros accesorios (pantalla, teclado, ratón) empleados para desarrollar la aplicación, se ha tenido en cuenta la tabla de periodos de amortización recogida por el plan general contable, que indica que los equipos para el proceso de la información con una vida útil en una empresa valorada a 4 años, se amortizan un 25% al año usando una cuota de amortización lineal. Se puede ver el resultado en la Tabla 4.

| **Equipo** | **Coste inicial** | **Amortización anual** | **Amortización durante el periodo de prácticas** |
|---|---|---|---|
| Ordenador | 800 € | 800€ x 0,25 = 200€ | 200€ / 1780 horas x 300 horas = 33,70€ |
| Accesorios | 330€ | 330€ x 0,25 = 82,5€ | 82,5€ / 1780 horas x 300 horas = 13,90€ |
| **Total** | | | **47,60€** |

Tabla 4: Costes asociados a los equipos materiales

A esto hay que añadir los costes del software utilizado, en este caso: Jira Software, Bitbucket, Adobe Illustrator, el curso de aprendizaje Android, así como la contratación de los servicios de Firebase y el coste de publicar la *app* en Google Play. Estos costes se ven en la Tabla 5.

No se han tenido en cuenta los costes proporcionales derivados del mantenimiento de la oficina, ya que debido al tamaño de la empresa es un coste mínimo frente a variaciones en el coste del salario.

En la Tabla 6 se han sumado los costes desglosados anteriormente dando un coste total de 5.614,71€.

| Software | Coste anual | Coste de 300h (periodo prácticas) |
|---|---|---|
| Jira | 105,47€ equipo de hasta 10 usuarios* | 105,47€ / 1780 horas x 300 horas = 17,78€ |
| Bitbucket | 0€ hasta 5 usuarios | 0€ |
| Adobe Illustrator | 290,28€ | 290,28€ / 1780 horas x 300 horas = 48,92€ |
| Curso Udemy | 10€ | 10€ |
| Firebase | 263,67€ con el plan Flame* [18] | 263,67€ / 1780 horas x 300 horas = 44,44€ |
| Google play | 21,97€ cuota de registro* | 21,97€ |
| **Total** | | **143,11€** |

Tabla 5: Costes asociados al software utilizado
(* Según tipo de cambio USD/EUR del 28 de junio de 2017 [19])

| Coste recursos humanos | Coste equipo | Coste software | COSTE TOTAL |
|---|---|---|---|
| 5.424€ | 47,60€ | 143,11€ | 5.614,71€ |

Tabla 6: Costes totales del proyecto

## 3.6 Seguimiento del proyecto

### 3.6.1 Periodo pre-sprints

Antes de comenzar con el desarrollo de los sprints, se dedicó un periodo de casi 3 semanas a la formación de la alumna en el desarrollo de aplicaciones con Android Studio. Para ello se realizó un curso online de aprendizaje Android, como ya se había mencionado anteriormente.

Esta formación ha sido muy importante a la hora de saber cómo plantear la aplicación, por dónde empezar a desarrollarla, así como para resolver conflictos básicos de la configuración con Android Studio.

Al acabar la formación se realizó un análisis básico de los requisitos de la aplicación, utilizando para ello las pantallas de diseño de interfaz de usuario proporcionadas por la empresa. A partir de

estas pantallas se establecieron las tareas necesarias, se formó la pila del producto, y se estimaron dichas tareas. Todo esto entró dentro del periodo estimado para la planificación inicial.

### 3.6.2 Primer sprint (21 de marzo – 4 de abril)

La estructura de los sprints siempre es la misma:

Se ha establecido el *backlog* del sprint, eligiendo 4 historias de usuario: HU01, HU03, HU04 y HU08, las cuales suman un total de 11 puntos de historia (3 + 2 + 3 + 3). Después se ha pasado al desarrollo y al final del sprint se ha hecho la revisión, comprobando que el resultado obtenido cumplía lo propuesto durante la planificación del sprint. En las Tablas 7 y 8 se puede observar el *backlog* del primer sprint, indicando las tareas que finalmente se completaron y las que no.

Como se observa, las historias de usuario del *backlog* han sido divididas en subtareas. Al principio se había decidido realizar completamente las historias adjudicadas al sprint, sin embargo, durante el desarrollo se decidió completar solamente la parte del *frontend*, y esperar al siguiente sprint para empezar a introducir la incorporación del *backend* con Firebase. Es por ello que se añadió la historia HU07, de 8 puntos, para completar en el tiempo no utilizado en el *backend*. Por tanto el sprint pasó de 11 a 19 puntos de historia.

| **Id historia** | **Descripción** | **Subtarea** | **Descripción** | **Completada al finalizar sprint** |
|---|---|---|---|---|
| HU01 | Registro de usuario | T01 | Hacer pantalla de registro | Sí |
| | | T02 | Programar funcionalidad registro | Sí |
| | | T03 | Integrar registro con servicio de autenticación Firebase | No |
| HU03 | Inicio de sesión | T04 | Creación pantalla selección de inicio | Sí |
| | | T05 | Programar funcionalidad selección inicio | Sí |
| | | T06 | Creación pantalla inicio de sesión | Sí |
| | | T07 | Programar funcionalidad inicio de sesión | Sí |
| | | T08 | Integrar inicio de sesión con servicio de autenticación Firebase | No |

Tabla 7: *Backlog* sprint 1 (I)

| Id | Descripción | Subtarea | Descripción | Completada al finalizar sprint |
|---|---|---|---|---|
| HU04 | Ver mi perfil | T09 | Creación pantalla de perfil de usuario | Sí |
| | | T10 | Programar funcionalidad perfil de usuario | Sí |
| | | T11 | Recoger datos perfil de base de datos Firebase | No |
| HU08 | Ver información apuestas | T12 | Creación del ítem de apuestas | Sí |
| | | T13 | Crear las clases del modelo necesarias para representar los ítems de apuestas | Sí |
| | | T14 | Crear funcionalidad para rellenar la información de los ítems | No |
| HU07 (historia añadida) | Listar apuestas | T15 | Creación pantallas populares, hoy, categorías, subcategorías | Sí |
| | | T16 | Programar funcionalidad | No |
| | | T17 | Recoger datos perfil de base de datos Firebase | No |

Tabla 8: *Backlog* sprint 1 (II)

Al no haberse realizado ninguna historia completa, no se pueden contar los puntos de historia que se han completado. Sin embargo, si tenemos en cuenta que la integración de Firebase representa aproximadamente el 30% de cada historia, y que la tarea 16, que no es de integración con Firebase, tampoco fue completada, se puede estimar que se completaron aproximadamente 11 puntos de historia de los 19 del sprint.

### 3.6.3 Segundo sprint (5 de abril – 27 de abril)

En la Tabla 9 se puede observar que para este sprint se han recuperado las tareas no realizadas del anterior (equivalentes a 8 puntos). Además, se han añadido dos tareas que no estaban presentes en el *backlog* inicial, la formación en Firebase, que se ha hecho mediante el estudio de la documentación proporcionada en la página oficial [20], estimada en 3 puntos de historia, y la realización de la base de datos noSQL en Firebase (2 puntos de historia). Al considerar la historia de configuración de notificaciones, se ha decidido revalorar su complejidad de 1 a 2 puntos de historia. El total de puntos asignados a este sprint ha sido de 15 puntos. Los puntos totales del proyecto con los incrementos explicados son 56.

| Id | Descripción | Subtarea | Descripción | Completada al finalizar sprint |
|---|---|---|---|---|
| T18 | Formación en Firebase | - | - | Sí |
| HU01 | Registro de usuario | T03 | Integrar registro con servicio de autenticación Firebase | Sí |
| HU03 | Inicio de sesión | T08 | Integrar inicio de sesión con servicio de autenticación Firebase | Sí |
| HU04 | Ver mi perfil | T11 | Recoger datos perfil de base de datos Firebase | Sí |
| HU08 | Ver información apuestas | T14 | Crear funcionalidad para rellenar la información de los ítems | Sí |
| HU07 | Listar apuestas | T16 | Programar funcionalidad | Sí |
| | | T17 | Recoger datos perfil de base de datos Firebase | Sí |
| HU06 | Configurar notificaciones | T19 | Crear pantalla de configuración | Sí |
| | | T20 | Programar funcionalidad de notificaciones | Sí |
| | | T21 | Registrar el *token* del dispositivo en la base de datos | Sí |
| T22 | Diseño base de datos | - | - | Sí |

Tabla 9: *Backlog* sprint 2

### 3.6.4 Tercer sprint (28 de abril – 12 de mayo)

Como se puede consultar en la Tabla 10, en este sprint se ha añadido las historias de usuario HU09 y HU10, de 5 puntos de historia cada una. También se han añadido tareas nuevas relacionadas con dichas historias, las tareas T33 y T34, cada una de 2 puntos de historia, y la tarea T23, de otros 2 puntos. El total de puntos asignados a este sprint es de 16, y se han completado 14. El total de puntos del proyecto es ahora de 62.

| Id | Descripción | Subtarea | Descripción | Completada al finalizar sprint |
|---|---|---|---|---|
| T23 | Añadir funcionalidad de búsqueda a los listados de apuestas | - | - | Sí |
| HU09 | Unirse a apuestas públicas | T24 | Crear pantalla apuestas | Sí |
| | | T25 | Crear pantalla emergente para realizar apuesta pública | Sí |
| | | T26 | Crear funcionalidad pantalla emergente pública | Sí |
| | | T27 | Crear funcionalidad pública en pantalla apuesta | Sí |
| | | T28 | Registrar el alta de la apuesta pública en Firebase | Sí |
| HU10 | Realizar apuestas privadas | T29 | Crear pantalla emergente para realizar apuesta privada | Sí |
| | | T30 | Crear funcionalidad pantalla emergente privada | Sí |
| | | T31 | Crear funcionalidad privada en pantalla apuesta | Sí |
| | | T32 | Registrar el alta de la apuesta privada en Firebase | Sí |
| T33 | Creación pantalla y funcionalidad de apuestas de modalidad un ganador | - | - | Sí |
| T34 | Creación pantalla y funcionalidad de apuestas modalidad por puntos | - | - | No |

Tabla 10: *Backlog* sprint 3

### 3.6.5 Cuarto sprint (15 de mayo – 26 de mayo)

En este sprint se ha vuelto a añadir la tarea T34 y se ha hecho una nueva, la T35. Cada una de ellas vale 2 puntos. Además se han incluido las historias HU17, HU18, HU02, HU11, HU12, HU13 y HU16, de 1, 3, 2, 2, 3, 1 y 2 puntos respectivamente. El sprint tiene asignados 18 puntos de historia. Ahora el total de puntos del proyecto es de 64. Se pueden observar estos datos en la Tabla 11.

| Id | Descripción | Subtarea | Descripción | Completada al finalizar sprint |
|---|---|---|---|---|
| T34 | Creación pantalla y funcionalidad de apuestas modalidad por puntos | - | - | Sí |
| T35 | Creación pantalla y funcionalidad de apuestas modalidad 1x2 | - | - | Sí |
| HU17 | Ver actividad amigos | T36 | Crear pantalla de sección muro | Sí |
| | | T37 | Programar funcionalidad muro | Sí |
| | | T38 | Recoger datos de la base de datos | Sí |
| HU18 | Recibir notificaciones | T39 | Crear funcionalidad notificaciones | Sí |
| | | T40 | Crear código de funciones Firebase | Sí |
| HU02 | Vincular cuenta con Facebook | T41 | Crear pantalla emergente Facebook | Sí |
| | | T42 | Crear pantallas de amigos Facebook | Sí |
| | | T43 | Programar funcionalidad pantalla con API Facebook | Sí |
| HU11 | Invitar amigos a apuestas | T44 | Crear pantalla de adición de contrincantes | Sí |
| | | T45 | Programar funcionalidad | Sí |
| | | T46 | Recoger amigos de la base de datos | Sí |
| HU12 | Consultar estado mis apuestas | T47 | Crear pantalla mis apuestas | Sí |
| | | T48 | Recoger apuestas de base de datos | Sí |
| HU13 | Aceptar o rechazar apuestas | T49 | Añadir funcionalidad de rechazo de apuestas en pantalla mis apuestas | Sí |
| HU16 | Agregar amigos | T50 | Crear pantallas sección Amigos | Sí |
| | | T51 | Programar funcionalidad sección amigos | Sí |
| | | T52 | Recoger lista usuarios de la base de datos | Sí |

Tabla 11: *Backlog* sprint 4

### 3.6.6 Quinto sprint (29 de mayo – 2 de junio)

Durante el quinto sprint se añadieron 3 historias, las cuales suman 6 puntos, y una tarea de corrección de errores y revisión del código añadida de 2 puntos. Además, se aumentó la complejidad de la HU05 de 2 a 3 debido a un error producido al modificar la imagen del usuario. La Tabla 12 se corresponde a este sprint. La suma de los puntos de historia del sprint es de 9, y el total de puntos final del proyecto es de 63.

| **Id** | **Descripción** | **Subtarea** | **Descripción** | **Completada al finalizar sprint** |
|---|---|---|---|---|
| HU05 | Modificar perfil | T53 | Creación pantalla de modificación de perfil | Sí |
| | | T54 | Programar funcionalidad modificación de perfil | Sí |
| | | T55 | Integrar con base de datos Firebase | Sí |
| HU14 | Consultar clasificación apuestas | T56 | Creación pantalla de clasificación | Sí |
| | | T57 | Programar funcionalidad | Sí |
| | | T58 | Recoger apuestas de la base de datos | Sí |
| HU15 | Consultar resultados de usuarios y respuestas correctas | T59 | Creación pantallas de resultados | Sí |
| | | T60 | Programar funcionalidad | Sí |
| | | T61 | Recoger resultados de la base de datos | Sí |
| T62 | Corrección de errores y revisión de código | - | - | Sí |

Tabla 12: *Backlog* sprint 5

### 3.6.7 Resumen de los sprints

Finalmente y como resumen del seguimiento, en la Tabla 13 se observa el desarrollo completo de los sprint realizados con la metodología Scrum. Como se aprecia, es una metodología ágil que da flexibilidad a la hora de modificar las tareas del proyecto, de forma que al final de cada ciclo, durante la revisión del sprint, se puede valorar el trabajo realizado para encontrar formas de mejorar el trabajo en la siguiente iteración.

| | **Puntos de historia asignados** | **Puntos de historia añadidos al *backlog* total** | **Puntos de historia finalizados** | **Puntos de historia restantes** |
|---|---|---|---|---|
| **Pila de producto inicial** | 50 | - | - | 50 |
| **Sprint 1** | 19 | 0 | 11 | 50 + 0 – 11 = 39 |
| **Sprint 2** | 15 | 6 | 15 | 39 + 6 – 15 = 30 |
| **Sprint 3** | 16 | 6 | 14 | 30 + 6 – 14 = 22 |
| **Sprint 4** | 18 | 2 | 18 | 22 + 2 - 18 = 6 |
| **Sprint 5** | 9 | 3 | 9 | 6 + 3 – 9 = 0 |

Tabla 13: Resumen del seguimiento de los sprints

## 3.7 Gestión de riesgos

Para asegurar la calidad de MyBets, hay que tener en cuenta los riesgos asociados a este proyecto. Debido a su naturaleza, estos no son demasiados. Por ejemplo, no hay riesgos destacables asociados a los servidores de la aplicación. Al ser Firebase un servicio de *backend* que proporciona escalabilidad y mantiene los servidores, no se producirá un número de peticiones que no se pueda manejar. Sí existen sin embargo riesgos asociados a:

- La calidad de la interfaz de usuario: Para evitar que esta sea deficiente, se han utilizado los *mockups* de la empresa, los cuales siguen principios de Material Design [9].
- Errores en el análisis: Estos son contrarrestados gracias al análisis realizado, así como con las reuniones de revisión con el propietario del producto, donde se verifica que la aplicación realiza lo que se espera.

# Capítulo 4

# Análisis y diseño del sistema

En este apartado se pretende analizar la aplicación de forma completa, para que al realizar a continuación el diseño se tenga en mente la posibilidad de que necesite ser expandida en el futuro añadiendo más tipos de apuestas o categorías.

Se desea que la aplicación sea lo más flexible posible, permitiendo realizar cambios en el *backend* sin que el usuario necesite actualizar a menudo la aplicación.

En las siguientes secciones se dan a conocer los requisitos de datos de la aplicación, la arquitectura del sistema y la estructura de la base de datos noSQL en Firebase.

## 4.1 Análisis del sistema

### 4.1.1 Requisitos de datos

MyBets necesita guardar tanto los datos personales de los usuarios como de los juegos y de las diferentes apuestas realizadas. En las Tablas 14 a 22 se pueden observar los requisitos de datos del sistema.

| **Requisitos de datos:** RD01 Usuario | |
|---|---|
| **Identificador**: RD01<br>**Nombre**: Usuario | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, nombre completo, nombre de usuario, id de Facebook, url de foto de perfil, número de puntos, email, contraseña, lista de *tokens* de dispositivos. | |
| **Detalles:**<br>• Id de Facebook: Solo en cuentas vinculadas con Facebook<br>• url de foto de perfil: Referencia al almacenamiento de Firebase<br>• lista de *tokens* de dispositivos: identificadores para que Firebase reconozca al usuario | |

Tabla 14: Requisitos de datos de Usuario

| **Requisitos de datos:** RD02 Partida | |
|---|---|
| **Identificador**: RD02<br>**Nombre**: Partida | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, nombre, bote de puntos, categoría, fecha de fin, modalidad, fecha publicación, subcategoría, estado resultados. | |
| **Detalles:**<br>• Tipo: “publica” o “privada” | |

Tabla 15: Requisitos de datos de Partida

| **Requisitos de datos:** RD03 Apuesta | |
|---|---|
| **Identificador**: RD03<br>**Nombre**: Apuesta | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, estado juego, estado invitación, puntos apostados, selección resultados. | |
| **Detalles:**<br>• estado de juego: “jugada” o “pendiente”.<br>• Estado invitación: “aceptada” o “rechazada”<br>• La selección de resultados variará según la modalidad de la apuesta. Si es a un ganador será una cadena, si es por puntos o 1x2, la selección de resultados será una lista | |

Tabla 16: Requisitos de datos de Apuesta

| **Requisitos de datos:** RD04 Actividad | |
|---|---|
| **Identificador**: RD04<br>**Nombre**: Actividad | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, fecha, mensaje, puntos ganados. | |

Tabla 17: Requisitos de datos de Actividad

| **Requisitos de datos:** RD05 Ítem | |
|---|---|
| **Identificador**: RD05<br>**Nombre**: Ítem | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, nombre, url imagen. | |
| **Detalles:**<br>• url imagen: opcional | |

Tabla 18: Requisitos de datos de Ítem

| **Requisitos de datos:** RD06 Enfrentamiento | |
|---|---|
| **Identificador**: RD06<br>**Nombre**: Enfrentamiento | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, contrincante 1, contrincante 2 | |

Tabla 19: Requisitos de datos de Enfrentamiento

| **Requisitos de datos:** RD07 Categoría | |
|---|---|
| **Identificador**: RD07<br>**Nombre**: Categoría | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, nombre, url imagen fondo, url icono | |

Tabla 20: Requisitos de datos de Categoría

| **Requisitos de datos:** RD08 Subcategoría | |
|---|---|
| **Identificador**: RD08<br>**Nombre**: Subcategoría | **Versión:** 2 |
| **Tipo:** Datos a mantener | |
| **Datos específicos a mantener:** id, nombre | |

Tabla 21: Requisitos de datos de Subcategoría

| **Requisitos de datos:** RD09 Resultado | |
|---|---|
| **Identificador**: RD08<br>**Nombre**: Resultado | **Versión:** 2 |
| **Tipo:** id, lista resultados | |
| **Datos específicos a mantener:** id, lista resultados | |
| **Detalles:**<br>• resultados: Los resultados variarán según la modalidad de la apuesta. Si es a un ganador será una cadena, si es por puntos o 1x2, la selección de resultados será una lista. | |

Tabla 22: Requisitos de datos de Resultado

## 4.2 Diseño de la arquitectura del sistema

Tal como dice la información extraída de la documentación de Google Cloud Platform [21], las implicaciones de construir un *backend* para una aplicación móvil son:

- Tener en cuenta el menor almacenamiento del dispositivo
- Permitir la sincronización de datos entre múltiples dispositivos
- Manejar datos *offline*
- Envío de notificaciones y mensajes
- Minimizar gasto de la batería

Siguiendo las directivas dadas por Google Cloud, se ha seleccionado para esta aplicación un patrón de diseño que utiliza la nube de Google para crear servicios de *backend* que cumplan esos requisitos. En la Figura 4 se puede ver un esquema de dicho patrón, en el cual se observa que cuando desde un terminal se realizan cambios que afectan a la base de datos (por ejemplo un usuario modifica su perfil, realiza una nueva apuesta, crea una partida privada, etc.), estos son inmediatamente sincronizados y reflejados en el resto de dispositivos, de forma que se observa la información en tiempo real. La validación de datos y la seguridad se manejan de forma especial, a través de unas normas declaradas usando la interfaz de usuario de Firebase. Al usar Firebase como único *backend*, existen limitaciones en la configuración, pero a cambio se obtienen muchos servicios.

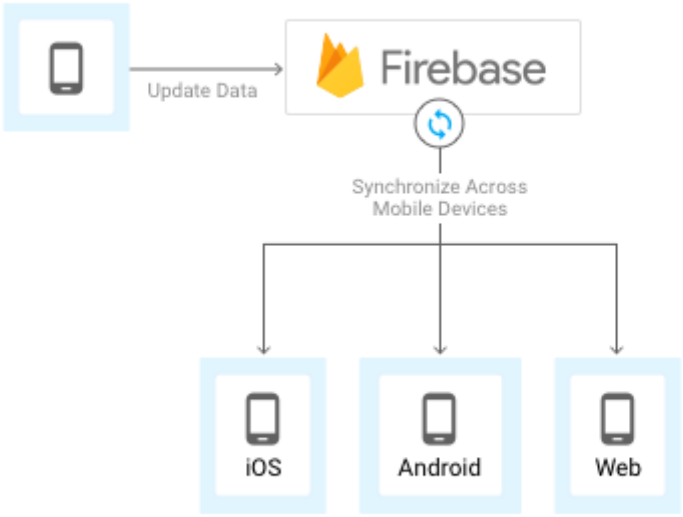


Figura 4: Arquitectura de la aplicación. Fuente: Google Cloud Platform [21]

Es por ello que la arquitectura de la aplicación se compone de dos partes:

- Una aplicación *frontend* diseñada para Android con el lenguaje Java
- Un *backend* en tiempo real realizado con Firebase

MyBets ha sido construida casi únicamente con código cliente, utilizando Firebase como *backend*. Esto ha sido posible debido a que la aplicación no necesita prácticamente ninguna integración con servicios de terceros. El único caso es el uso de la autenticación con Facebook, la cual también es manejada con la autenticación de Firebase. Sin embargo, sí ha sido necesario utilizar una función de Firebase para escribir un mínimo de código *backend* en Javascript, para poder realizar el reparto de los puntos a los usuarios una vez finalizadas las apuestas, así como para mandar las notificaciones cuando se producen ciertos eventos en el sistema. Pero todo esto puede realizarse con Cloud Functions, una parte de la plataforma Firebase que permite extender el código *backend* personalizado sin necesidad de administrar ningún servidor personalmente. Las funciones se disparan a partir de cambios producidos en la base de datos de Firebase.

Como se explica en su página principal [11], Firebase es un *backend* muy potente que proporciona las siguientes funcionalidades:

- Sincronización de datos en tiempo real: La base de datos noSQL de Firebase proporciona almacenamiento de datos sincronizados entre usuarios en tiempo real.
- Escalabilidad de los datos: Firebase proporciona un *backend* escalable horizontalmente que elimina la necesidad de mantener servidores.
- Menos código a mantener: Al no tener que realizar a penas código en la parte del *backend*, el programador puede centrarse en mejorar la experiencia del cliente, y el tiempo de desarrollo se reduce.
- Funciones *backend*: Para los pequeños casos en los que es necesario, Firebase permite la ejecución de funciones desarrolladas con Javascript, que se disparan al realizarse un cambio en la base de datos. Esto se ha usado principalmente para las notificaciones que llegan a los dispositivos de los usuarios.
- Administración de usuarios: En lugar de mantener la sesión en el dispositivo, cada petición realizada a Firebase contiene la información necesaria para que el dispositivo sea reconocido por Firebase y se pueda identificar al usuario. Esto se realiza guardando los *tokens* (identificadores) del dispositivo. Además, cuando la seguridad del *token* se considera comprometida, este se actualiza automáticamente. Al poder identificar de esta forma a los usuarios, Firebase autoriza el acceso a la información según las normas definidas en la propia base de datos.
- Almacenamiento de archivos: Las imágenes estáticas de MyBets, como el icono de la aplicación, se guardan localmente. Por otro lado, las imágenes dinámicas que dependen de qué consulta se esté realizando a la base de datos, como ver la imagen de perfil de un usuario o los iconos de las categorías, son guardadas en el almacenamiento de Firebase. Este servicio proporciona una url de descarga para cada una de ellas, y estas url se guardan en la base de datos. Por ejemplo, cuando un usuario modifica su imagen de perfil, la nueva

imagen se sube al almacenamiento de Firebase, y su url de descarga se guarda en el nodo que almacena los datos de dicho usuario. De esta forma, cuando el usuario intente ver su perfil, se recogerán sus datos de la base de datos, entre los cuales estará la url de descarga, que será usada para mostrar la imagen.

Mybets tiene una arquitectura de dos niveles, donde la comunicación entre el dispositivo Android y Firebase se realiza a través del SDK de Firebase, el cual usa *web sockets* para comunicarse con la base de datos, utilizando la técnica de *long polling*, mediante la cual el servidor Firebase mantiene la petición del cliente abierta hasta que se produce el cambio, momento en el que devuelve la información.

Es decir, cuando el dispositivo cliente se conecta a la base de datos, se abre una conexión *web socket* desde el dispositivo al servidor Firebase. Al enlazar un escuchador, el cliente le manda al servidor la localización de este, de forma que Firebase añade el escuchador a una lista de escuchadores de los clientes conectados. Cuando se realiza una operación de escritura en la base de datos, el servidor de Firebase manda una actualización a los escuchadores de su lista que han sido afectados a través del *web socket*.

Sin embargo, todo esto se sitúa en una capa oculta inferior, ya que en envío y recepción de datos se programa en el cliente a través de la API de Firebase.

## 4.3 Diseño de la base de datos

Como se ha explicado, se ha utilizado una base de datos noSQL para guardar los datos de la aplicación. Tal y como se explica en la página de los productos que ofrece Firebase [22], el almacenamiento de datos sincronizados de Firebase permite que se pueda acceder a dicha base de datos desde cualquier dispositivo, ya sea móvil o web. Cuando se actualizan los datos desde un dispositivo, estos se guardan en la nube de Google y se notifica simultáneamente a los usuarios pertinentes en periodos de milisegundos. También es posible el uso *offline* de la base de datos. Si un usuario pierde su conexión, el SDK guarda los cambios en una caché local. De esta forma, cuando se recupera la conexión, los datos locales se sincronizan de forma automática.

Firebase también nos informa de que para mantener la seguridad de la base de datos, hay normas de seguridad que permiten especificar qué usuarios pueden acceder a qué datos y cómo se debe estructurar la base de datos. Estas normas se administran en la interfaz de usuario en la web de Firebase.

La principal ventaja de esta base de datos es que al estar alojada en la nube de Google no hay necesidad de mantener servidores, y el SDK de Android permite que la sincronización sea fácil de implementar, utilizando las funciones que Firebase proporciona.

Al ser una base de datos noSQL, es importante estructurar bien los datos. Para ello se han seguido las directivas proporcionadas por Firebase en su documentación [23], donde informan de que todos los datos de Firebase Realtime Database se guardan en formato JSON. Es decir, que no existen tablas ni registros para guardar los datos, y cada vez que se agregan datos al árbol JSON, estos se convierten en un nodo en la estructura JSON existente. También se comenta en la

documentación que es importante que los datos no tengan demasiados niveles de anidación, ya que cuando se permite el acceso de lectura o escritura en la base de datos, se le otorga también acceso a los nodos que hay por debajo. Por eso es mejor mantener la estructura simple aunque a cambio se produzca repetición de información.

En el anexo B se muestra cómo se ha estructurado la base de datos de MyBets utilizando la interfaz de usuario de Firebase. Como se ha dicho, estos datos se guardan en un árbol JSON, sin embargo la interfaz de Firebase permite ver de un vistazo la estructura completa. Hay que tener en cuenta que las figuras del anexo resumen la base real utilizada, ya que esta última tiene muchos más nodos de los mostrados.

La función de cada nodo de la rama principal de la aplicación es la siguiente:

- **actividadPerfil:** Nodo que almacena las últimas actividades del usuario. Se organiza según el *id* de usuario.
- **amigos:** Nodo que almacena la lista de identificadores de los amigos de cada usuario.
- **apuestas:** Nodo que almacena los datos de cada apuesta realizada, y los identificadores de las partidas, usuarios y resultados seleccionados.
- **competicionesSeguidas:** Guarda los *tokens* de dispositivos que serán informados cuando se tengan los resultados de una partida.
- **enfrentamientos:** Guarda las listas de datos de enfrentamientos de diferentes competiciones deportivas.
- **categorias:** Guarda los datos de las diferentes categorías de apuestas.
- **items:** Lista de las posibles elecciones de resultados de cada partida.
- **muro:** Listado de la actividad de los amigos de un usuario.
- **partidasPrivadas:** Datos de las partidas privadas creadas por los usuarios.
- **partidasPublicas:** Datos de las partidas públicas de la aplicación.
- **resultados:** Selección de resultados de cada apuesta hecha por los usuarios.
- **resultadosCorrectos:** Resultados finales correctos de las partidas.
- **subcategorias:** Datos de las subcategorías de apuestas.
- **tokens:** Lista de identificadores de los dispositivos de cada usuario.
- **usuarios:** Datos de los usuarios de la aplicación.

## 4.4 Diagrama de clases

Como se explica en el apartado 4.3, se ha utilizado una base de datos noSQL para almacenar los datos de la aplicación. Es decir, la información se ha guardado en nodos independientes, y no tablas relacionales. Sin embargo, las lecturas y escrituras a los nodos sí se hacen muchas veces a partir de información previa extraída de otros nodos. Por ejemplo, cuando un usuario participa en una apuesta, en el nodo de esta apuesta queda guardado el identificador del usuario que la realizó. Sí podemos por tanto, realizar un esquema que represente la estructura de información del sistema de forma que se alcance una visión global de la información más fácilmente que si solo contemplamos la estructura JSON de la base de datos. El esquema utilizado es el diagrama de clases representado en la Figura 5.

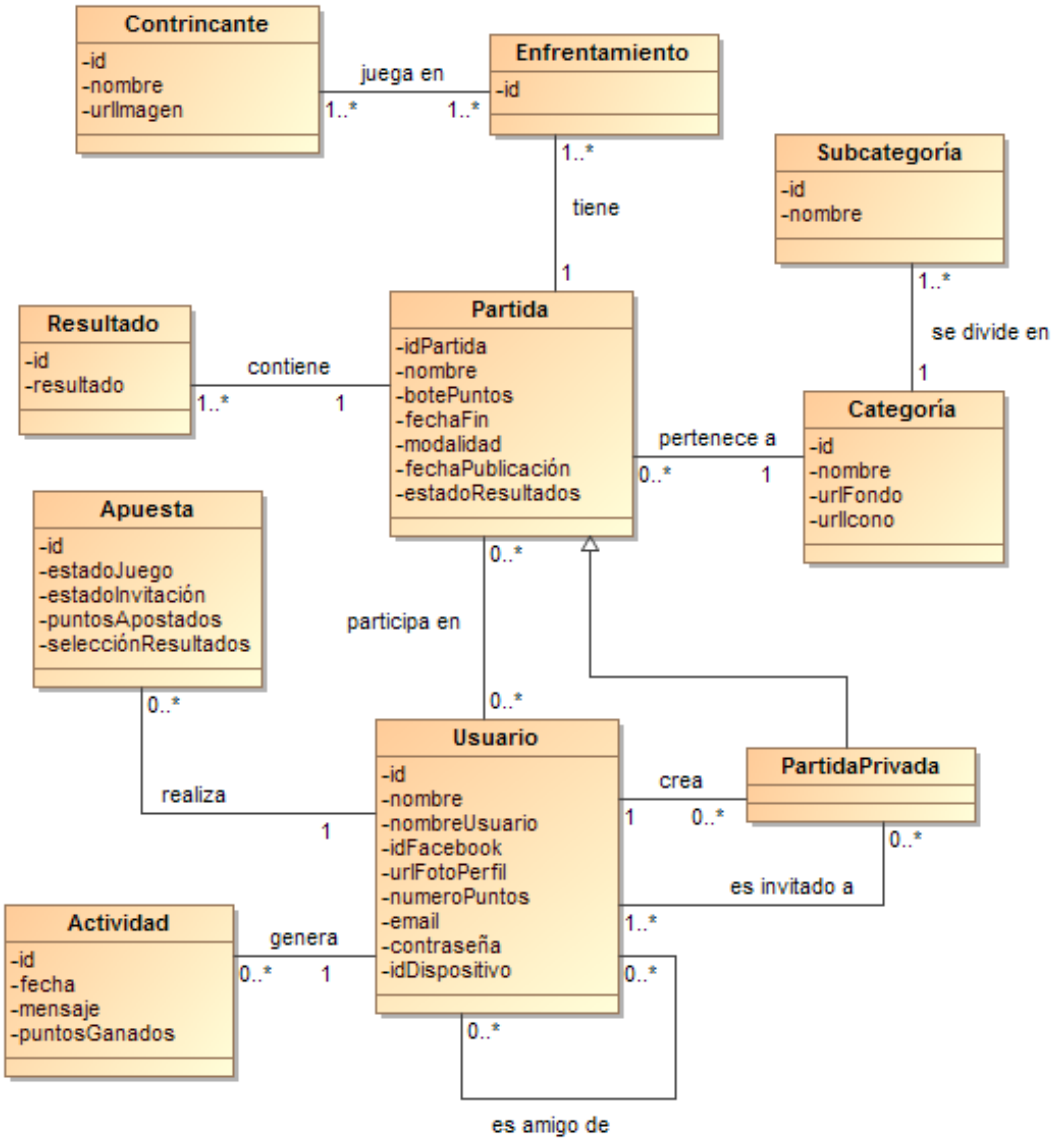


Figura 5: Diagrama de clases de MyBets

## 4.5 Diseño de la interfaz

La empresa Cuatroochenta ha proporcionado los recursos de la aplicación en forma de *mockups*. El equipo de diseño ha sido el encargado de dictaminar el aspecto visual de esta, de forma que se cumplan los principios de Material Design de Android [9]. La lista completa de los *mockups* proporcionados por Cuatroochenta puede consultarse en el anexo C. A continuación se detalla el aspecto final de la aplicación tal y como ha sido implementada en este proyecto, explicando las diferencias aplicadas respecto a los *mockups* propuestos. Las figuras 6 a 39 son capturas de pantalla de la aplicación en funcionamiento.

En la Figura 6 se puede ver el *splash* de la aplicación. Esto es lo primero que ve el usuario cuando abre la aplicación sin tener la sesión iniciada. En esta pantalla se muestra el logo de la aplicación, y el usuario tiene las opciones de iniciar sesión con Facebook, iniciar con su email y contraseña, lo cual llevaría a la Figura 7, o registrarse, que le llevaría a la Figura 8. En la Figura 9 hay un ejemplo de la validación al registrarse en la aplicación. Una vez el usuario se registra, inicia sesión automáticamente.

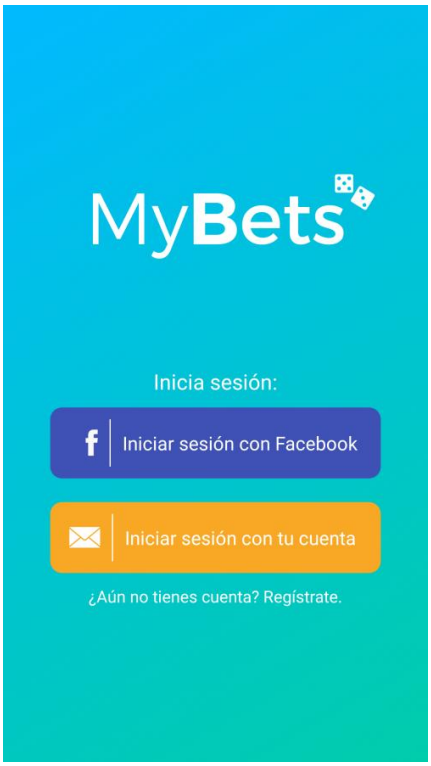


Figura 6: Pantalla selección de inicio


Figura 7: Pantalla de inicio de sesión

Una vez que el usuario haya iniciado sesión, se le redirigirá a la pantalla de las Figuras 10 y 11. Si el usuario no cierra la sesión, la próxima vez que el usuario abra la aplicación se le dirigirá a esta pantalla, la cual es la principal de MyBets. En ella se incluye información de los puntos del usuario, tres botones que redirigen a sus correspondientes pantallas: Categorías (Figura 12), Populares (Figuras 13 y 14) y Fútbol, que es un acceso a la pantalla de categoría de deportes (Figura 15) mostrando solo apuestas de fútbol. A continuación se muestra una lista deslizable verticalmente con tres de las apuestas publicadas en el día actual. Si se pulsa el texto "MÁS", se dirige al usuario a la pantalla de la Figura 16, donde se muestran todas las apuestas publicadas en el día actual. Además, como se observa en la Figura 11, también se dispone un botón con un menú desplegable para acceder a la configuración (Figura 17), al perfil (Figura 18) o cerrar sesión, lo cual devolvería al usuario a la pantalla de selección de inicio de sesión.

La pantalla de selección de categorías (Figura 12) muestra una lista de categorías. Clicando en ellas se dirige al usuario al listado de las diferentes categorías de apuestas (Figura 15). La pantalla Populares (Figura 13) muestra la lista de apuestas ordenadas según número de participantes de mayor a menor.

La Figura 14 muestra un ejemplo de búsqueda en el listado de apuestas. Si analizamos en esta pantalla la información que muestran las apuestas, se puede observar:

- Una imagen que cambia de color y de icono dependiendo la categoría de la apuesta, en este caso es morada con la imagen de un sobre, representando la categoría de política.
- El nombre y la modalidad de la apuesta: "Francia 2017 (un ganador)".
- Un número en la esquina superior derecha que indica el número de participantes actual.
- Debajo del nombre se encuentra la fecha de finalización, pero como en este caso la apuesta acaba en el día actual se muestra en su lugar un temporizador que se actualiza cada segundo.
- Los puntos del bote de la apuesta.
- Un botón "Juega ya" para participar en la apuesta.

Por otro lado, en la Figura 17 observamos la pantalla de configuración, la cual tiene un botón de tipo interruptor para que el usuario elija si desea o no recibir notificaciones en su teléfono.

En la Figura 18 se ve el perfil de usuario, donde se muestra su nombre, imagen de perfil, puntos y actividad reciente. También hay un botón con forma de lápiz que redirige a la pantalla de edición de perfil (Figura 19). En esta última se puede editar los datos del usuario, incluyendo la imagen de perfil. Cuando se clica en la imagen, la aplicación deja elegir una nueva de la galería de fotos del dispositivo.

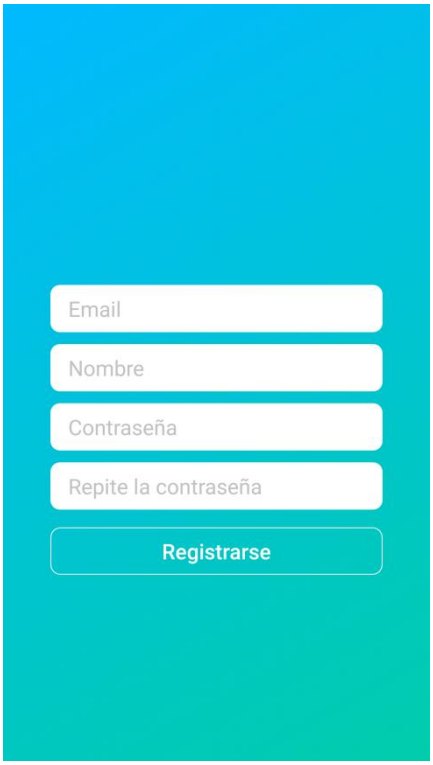


Figura 8: Pantalla de registro (I)

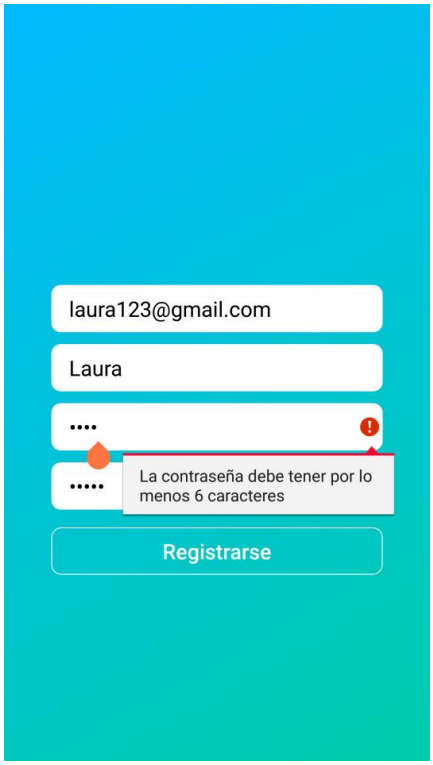


Figura 9: Pantalla de registro (II)


Figura 10: Pantalla principal (I)

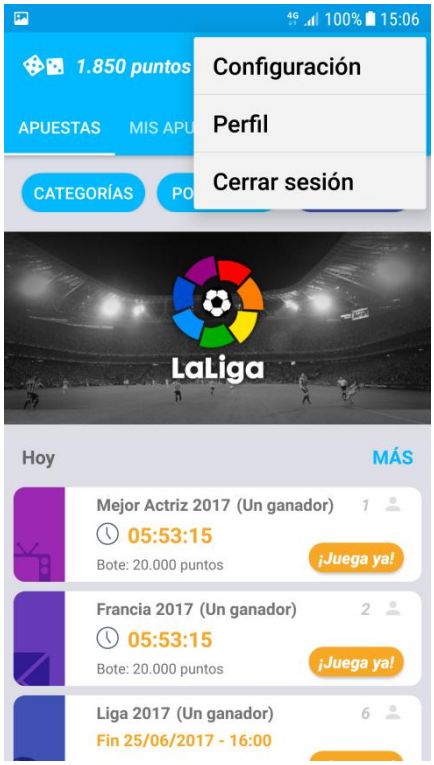


Figura 11: Pantalla principal (II)

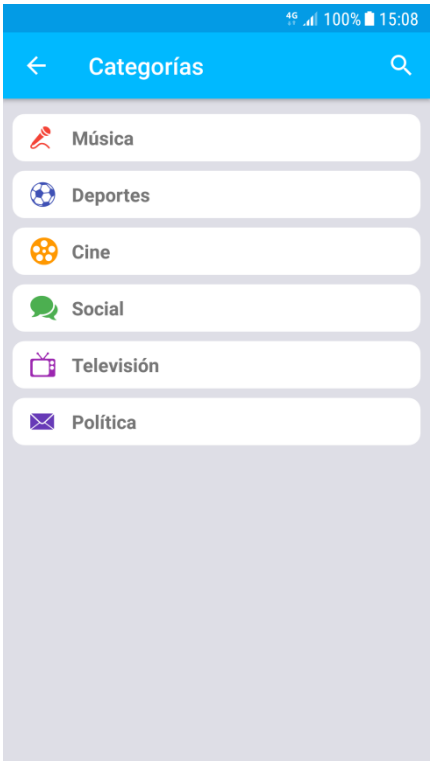


Figura 12: Pantalla selección de categoría

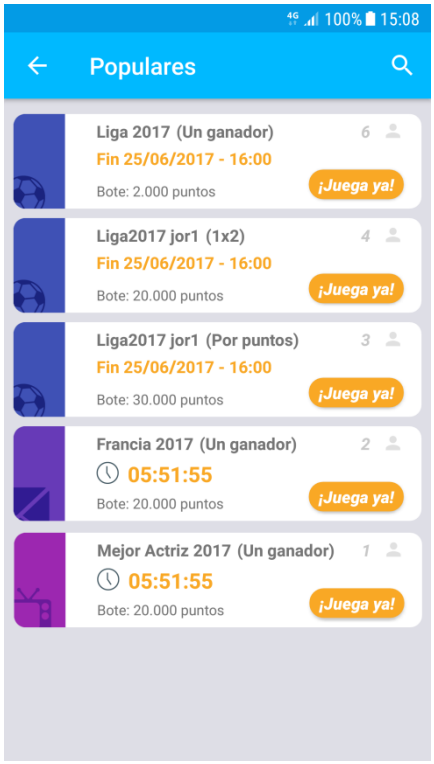


Figura 13: Pantalla apuestas populares (I)

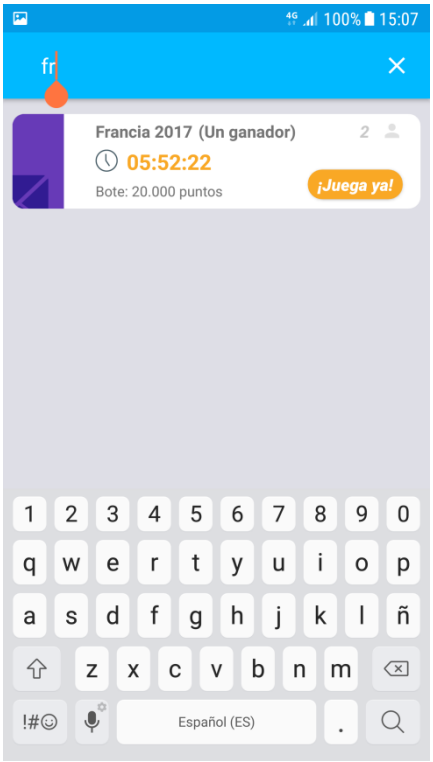


Figura 14: Pantalla apuestas populares (II)

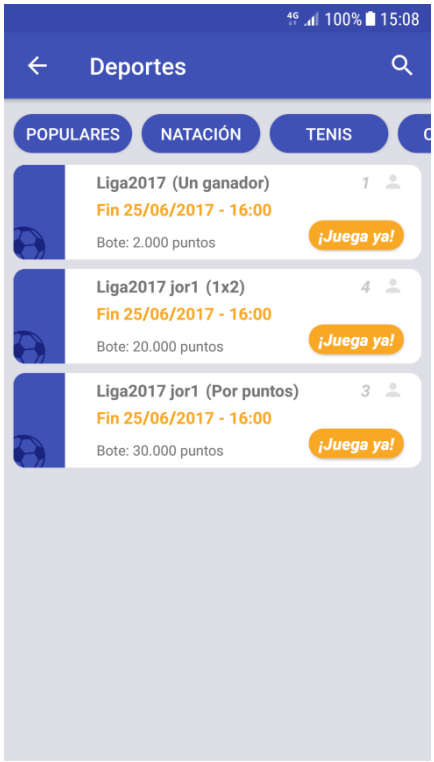


Figura 15: Pantalla categoría deportes


Figura 16: Pantalla apuestas del día actual

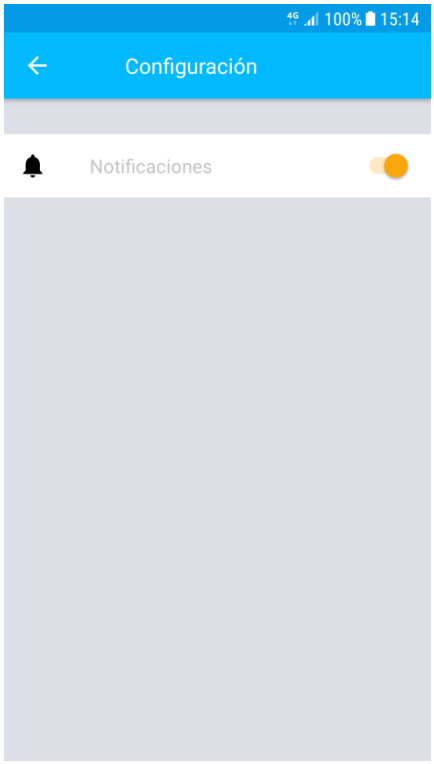


Figura 17: Pantalla de configuración


Figura 18: Pantalla de perfil

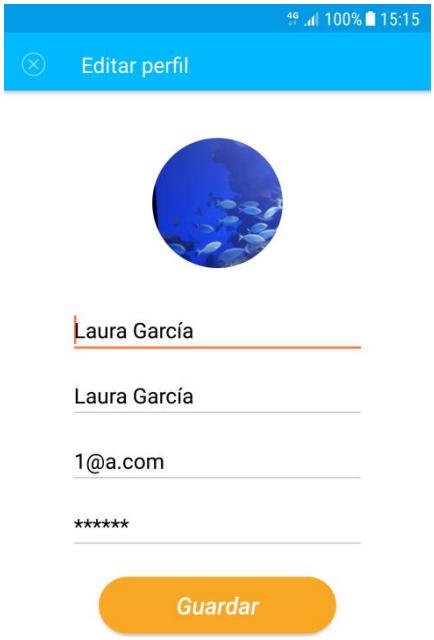


Figura 19: Pantalla de edición de perfil

Para conocer el estado de las apuestas realizadas por el usuario, así como aquellas a las que ha sido invitado, hay que ir a la sección "Mis Apuestas" del menú principal (Figura 20). En esta sección se puede ver que las apuestas se dividen en dos listas deslizables diferentes:

- Pendientes: Las apuestas que todavía están pendientes de recibir los resultados finales de la aplicación.

- Pasadas: Las apuestas que ya disponen de los resultados finales.

Si además se observa la información de las apuestas, se puede ver que se muestra de forma diferente respecto al resto de secciones. La imagen de fondo, nombre, modalidad y fecha se representan de forma similar, sin embargo se observan los siguientes cambios:

- El número que representaba los participantes de la partida, en las apuestas que son privadas se representa de la siguiente forma: "1/2" siendo la primera cifra el número de participantes, y la segunda el número de invitados a participar en la partida privada. Es decir, en una apuesta donde el número es "1/2" significa que de un total de 2 jugadores invitados, ha jugado en la apuesta 1 hasta el momento.

- En lugar del bote de puntos, se muestra si el usuario ha apostado o no, y si lo ha hecho, el número de puntos apostados.

- El botón "Juega Ya" desaparece una vez se rechaza o se juega a una partida. Si se ha jugado, aparece el texto "Esperando resultados", si se rechaza una partida a la que el usuario ha sido invitado, aparece el texto "Apuesta rechazada".

En la Figura 21 se puede ver el desplegable para rechazar una invitación a participar en una partida.

La Figura 22 muestra la pantalla del muro, donde el usuario puede consultar las últimas actividades de sus amigos, agrupadas por fecha.

En la Figura 23 se representa la sección de amigos, donde se muestran los amigos del usuario ordenados por el más recientemente agregado. El botón flotante de la esquina inferior derecha redirige a la pantalla de añadir amigos (Figura 24). Si en esta última pantalla se clica en la barra "Contactos de Facebook", se redirige a dos pantallas diferentes:

- Si el usuario no ha iniciado sesión con una cuenta de Facebook o con una cuenta de correo vinculada a una cuenta de Facebook, se muestra el diálogo de la Figura 25. Clicando en "Iniciar sesión con Facebook", se dejará al usuario iniciar sesión en Facebook y se vincularán las dos cuentas. Después se le dirigirá a la pantalla de amigos de Facebook (Figura 26).

- Si el usuario ha iniciado sesión con una cuenta de Facebook o una cuenta de correo vinculada a una cuenta de Facebook, se le dirige a la pantalla de amigos de Facebook (Figura 26), donde se muestran los contactos de Facebook que también han usado la aplicación.


Figura 20: Pantalla Mis Apuestas (I)


Figura 21: Pantalla Mis Apuestas (II)

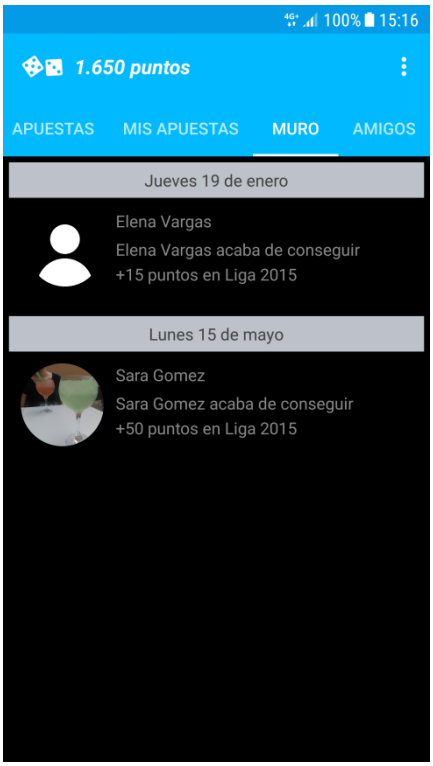


Figura 22: Pantalla Muro

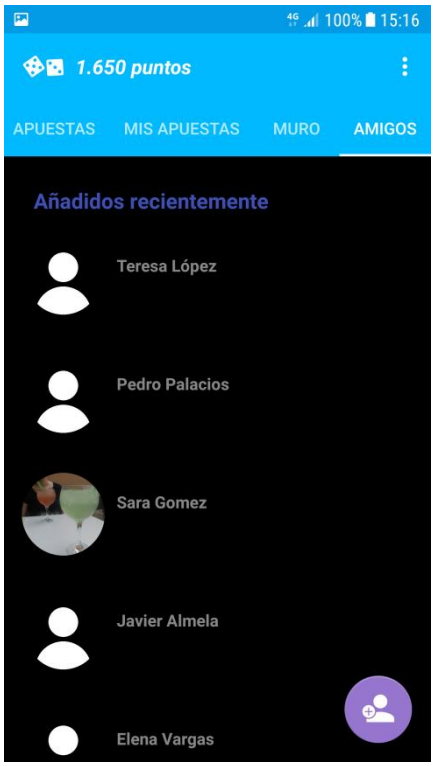


Figura 23: Pantalla Amigos

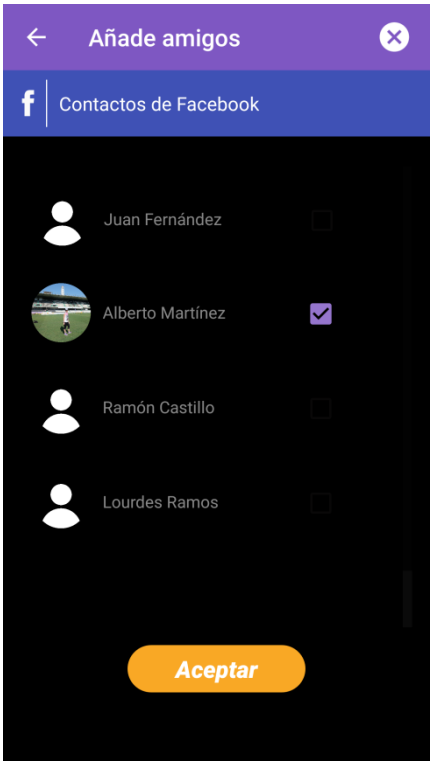


Figura 24: Pantalla de añadir amigos


Figura 25: Diálogo de vinculación a Facebook

La pantalla de la Figura 27 muestra el formato de las notificaciones recibidas en el dispositivo. En este caso se ha recibido una notificación producida cuando un usuario ha agregado a otro como amigo.

Cuando en una pantalla que muestra la información de una partida, se clica en el botón "Juega ya", se muestra un diálogo donde el usuario puede introducir los puntos que quiere gastar, y dos botones para elegir si quiere unirse a la apuesta pública, o crear un juego privado (Figuras 28 y 29). Si el usuario decide unirse a una partida pública, se le mandará a la pantalla de selección de resultados, la cual variará según la modalidad de la partida. Si por el contrario quiere hacer un juego privado, antes se le dirigirá a la pantalla de selección de oponentes (Figura 30).

Hay tres tipos de modalidades de partida:

- Un ganador: Partidas en las que se selecciona un único resultado, el ganador de la partida.
- 1x2: Partidas en las que el jugador ha de seleccionar un resultado para cada enfrentamiento de una lista. Los posibles resultados a elegir son "1" (gana el primer equipo), "2" (gana el segundo equipo) o "X" (los equipos empatan).
- Por puntos: Partidas en las que el jugador ha de escribir un resultado para cada enfrentamiento de una lista. Los resultados se introducen como número enteros, y representan los puntos que anota cada contrincante del enfrentamiento.

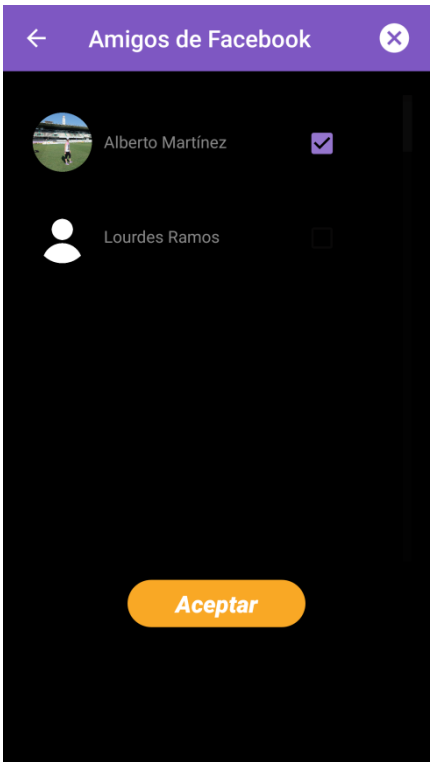


Figura 26: Pantalla Amigos de Facebook

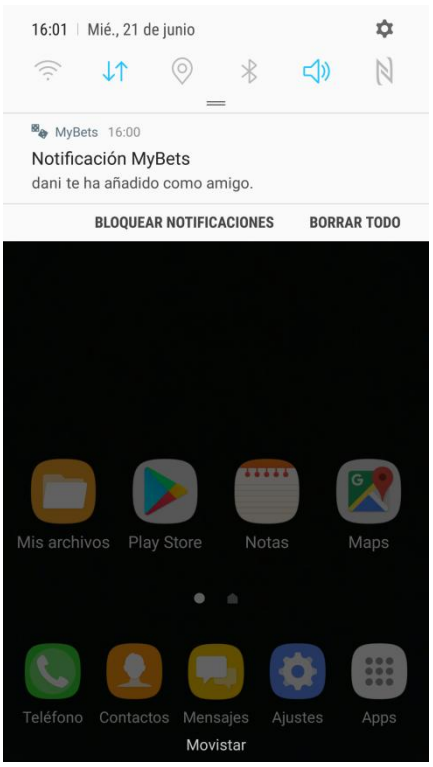


Figura 27: Notificación


Figura 28: Diálogo para apostar (I)


Figura 29: Diálogo para apostar (II)

En la Figura 31 se muestra un ejemplo de realización de una apuesta a un ganador, en las Figuras 32 y 33 de una partida 1x2 y en la Figura 34 de una partida por puntos. Además en la Figura 33 se puede ver que cuando se intenta apostar antes de seleccionar todos los resultados, se nos muestra un mensaje de error.

En la Figura 35 se puede observar el diálogo que aparece cuando se envía una apuesta.

Por último, falta hablar de las pantallas de clasificación. Cuando desde la sección "Mis Apuestas" de la aplicación, se pulsa una apuesta de la lista de "Pasadas", es decir, aquellas que ya disponen de resultados, se dirige al usuario a la pantalla de clasificación (Figura 36), donde se puede consultar qué usuarios han ganado más puntos en dicha partida. Al clicar sobre los usuarios de la lista de clasificados, según la modalidad de la partida se dirige a los usuarios a las pantallas de las Figuras 37, 38 y 39.

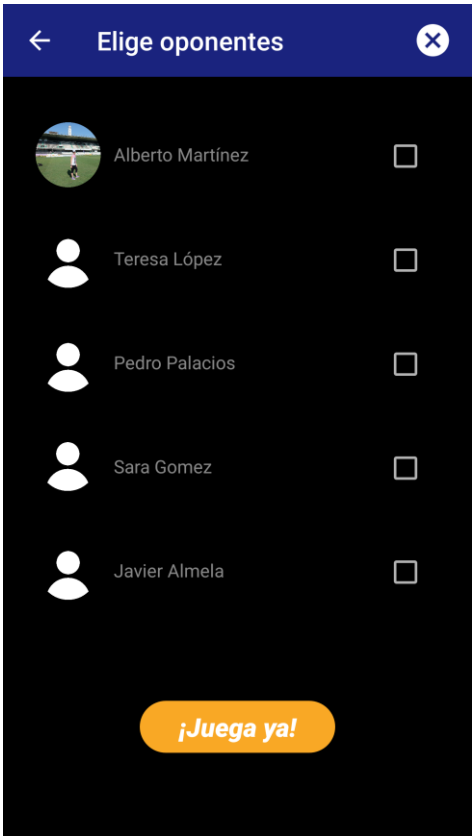


Figura 30: Pantalla de selección de contrincantes

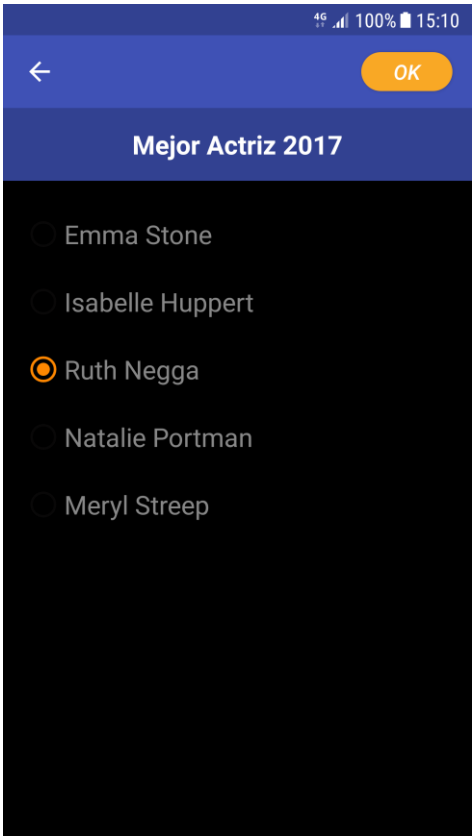


Figura 31: Pantalla de apuesta a un ganador

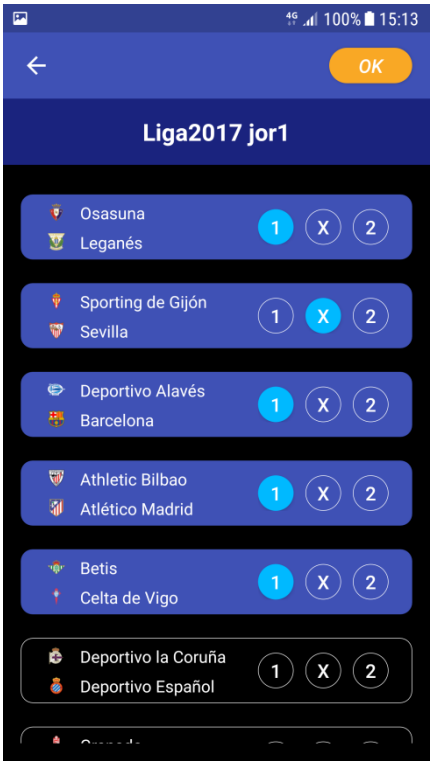


Figura 32: Pantalla de apuesta 1x2 (I)


Figura 33: Pantalla de apuesta 1x2 (II)

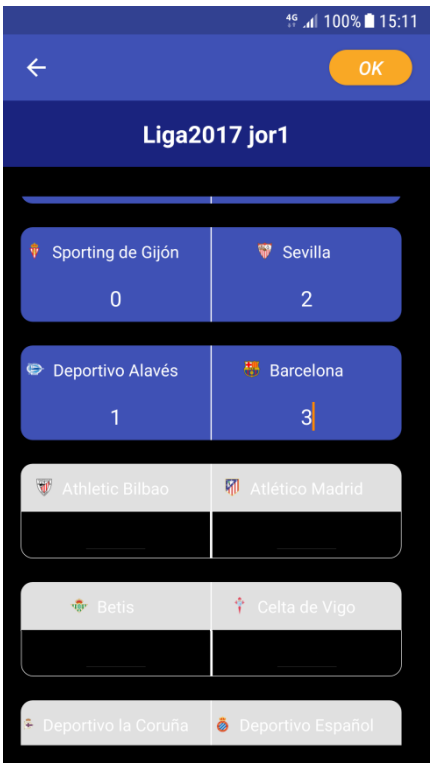


Figura 34: Pantalla de apuesta por puntos (I)


Figura 35: Diálogo de apuesta enviada

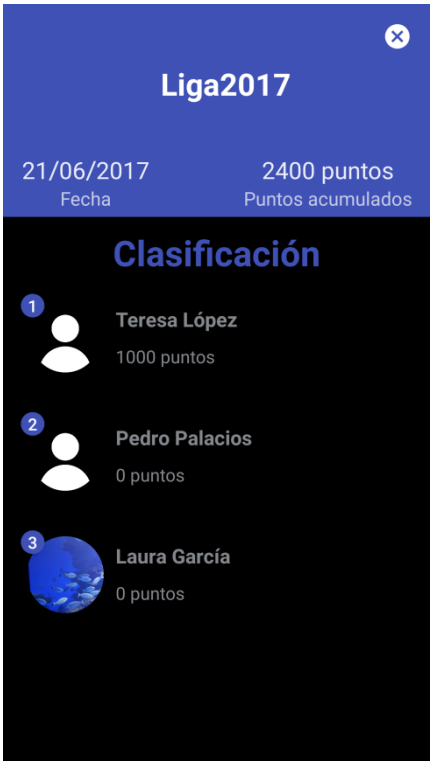


Figura 36: Pantalla de clasificación


Figura 37: Pantalla de resultados un ganador

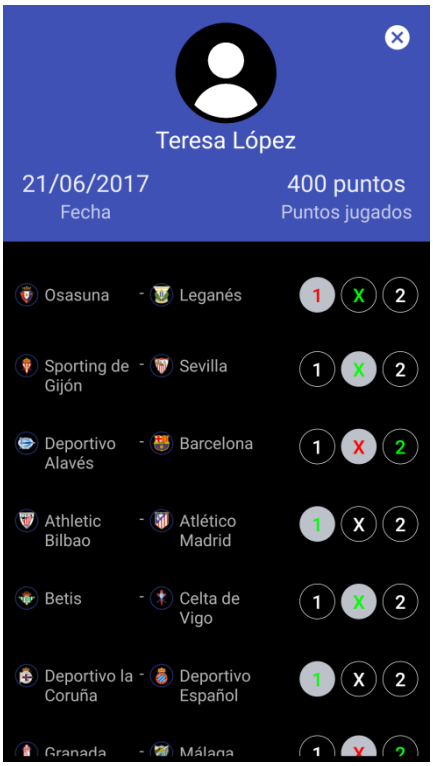


Figura 38: Pantalla de resultados 1x2


Figura 39: Pantalla de resultados por puntos

# Capítulo 5

# Implementación y pruebas

## 5.1 Detalles de implementación

### 5.1.1 Componentes de la aplicación

Los componentes clave en una aplicación Android son las actividades, que tal y como se definen en la guía de desarrolladores Android [24], definen la funcionalidad de las pantallas de la interfaz de usuario. Además, aunque pueden relacionarse pasándose información y llamándose unas a otras, las actividades son independientes entre sí. Para implementar una actividad, se debe escribir una clase Java que hereda de *Activity*. En el extracto de código de MyBets mostrado en la Figura 40, se puede ver cómo la clase usada hereda de *AppCompatActivity*, que es una clase que además de la funcionalidad de *Activity* ofrece funciones extra de compatibilidad con versiones de Android anteriores.

La guía de desarrolladores también explica que las llamadas a los componentes se hacen mediante mensajes asíncronos, los cuales enlazan componentes en tiempo de ejecución. Además, se puede pasar información de un componente a otro almacenándola en los mensajes. En la Figura 41 puede verse un ejemplo de esto. En la línea 20 se instancia el mensaje, llamado *intent*, y se le añade la información en la línea 21. La actividad se inicia en la línea 22 con la función *startActivity (intent).*

Para que los componentes se puedan iniciar hay que declararlos en el *Android Manifest*, un archivo XML de la aplicación. Cuando se instala una aplicación Android en un dispositivo, se pide al usuario los permisos necesarios. En el caso de Mybets, los permisos requeridos son de conexión a Internet y acceso al estado de la red. Estos permisos se configuran en el archivo *manifest.xml* de la aplicación (líneas 5 y 6 de la Figura 42).

Además del código de la aplicación, en MyBets se han usado imágenes como recursos y se han definido los estilos y colores de cada pantalla de la aplicación. Los estilos de las pantallas se definen en archivos XML, tal y como se ve en la Figura 43.

```
10
11  public class BaseActivity extends AppCompatActivity {
12
13      public DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();
14
15      @VisibleForTesting
16      public ProgressDialog mProgressDialog;
17
18      public void showProgressDialog() {
19          if (mProgressDialog == null) {
20              mProgressDialog = new ProgressDialog(this);
21              mProgressDialog.setMessage("Cargando...");
22              mProgressDialog.setIndeterminate(true);
23          }
24
25          mProgressDialog.show();
26      }
27
28      public void hideProgressDialog() {
29          if (mProgressDialog != null && mProgressDialog.isShowing()) {
30              mProgressDialog.dismiss();
31          }
32      }
33
```

Figura 40: Código de MyBets que muestra la herencia de *AppCompatActivity*

```
13      @Override
14      public void verRanking(Apuesta apuesta, Partida partida) {
15          if (apuesta.isEstadoResultados()) {
16              Bundle b = new Bundle();
17              b.putString("tipoPartida", apuesta.getTipo());
18              b.putString("puntos", ((MainActivity) getActivity()).getUsuario().getPuntos());
19              b.putString("idPartida", apuesta.getIdPartida());
20              Intent intent = new Intent(getApplicationContext(), ClasificacionActivity.class);
21              intent.putExtras(b);
22              startActivity(intent);
23          }
24      }
```

Figura 41: Código de MyBets que muestra la redirección entre actividades

### 5.1.2 Estructura del proyecto

Al configurar Android Studio y crear el proyecto, este separa automáticamente las clases Java de los recursos formados por ficheros XML e imágenes. Las clases Java se han dividido a su vez en aquellas que proporcionan parte de la interfaz de usuario y las que forman parte del modelo de datos y sirven para guardar información de objetos recibidos en formato JSON desde la base de datos. Si se estudia la Figura 44, se puede ver el árbol de la estructura del proyecto en Android Studio.

El código desarrollado se encuentra dentro de la carpeta “main”. Esta se divide en dos carpetas más: “java” y “res”. Tal como indica su nombre, en la primera se encuentra todo el código desarrollado en Java. Las clases están organizadas según si son *activities* que controlan pantallas de

la aplicación (carpeta “controller”), clases con métodos que realizan operaciones de ayuda (carpeta “funcionesDeAyuda”), clases del modelo o clases de servicios. Por otro lado, en la carpeta “res” encontramos los archivos XML donde se diseñan las pantallas (carpetas “layout” y “layout-land”), así como las imágenes y recursos dibujados en archivos XML (carpetas con nombre que empieza por “drawable” o “mipmap”). En las carpetas que empiezan por “values” se encuentran los XML con valores de constantes y estilos de diseño utilizados en el resto de archivos de diseño, y en la carpeta “xml” hay archivos de preferencias de las pantallas de configuración y de la búsqueda de la aplicación.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.marinacastellote.mybetsb">

    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon_launcher"
        android:label="MyBets"
        android:roundIcon="@drawable/icon_launcher"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <meta-data android:name="com.facebook.sdk.ApplicationId" android:value="1844702359113457"/>
        <meta-data
            android:name="com.google.firebase.messaging.default_notification_icon"
            android:resource="@drawable/noti_dados" />
        <meta-data
            android:name="com.google.firebase.messaging.default_notification_color"
            android:resource="@color/colorPrimary" />
        <activity
            android:name=".controller.anyadirAmigos.AnyadirAmigosActivity"
            android:parentActivityName=".controller.paginaPrincipal.MainActivity"
            android:label="MyBets"
            android:screenOrientation="portrait"
            android:theme="@style/NoStatusBar">
            <meta-data android:name="android.app.searchable" android:resource="@xml/searchable_config"/>
        </activity>
        <activity
            android:name=".controller.anyadirAmigos.AnyadirAmigosFacebook"
            android:label="MyBets"
            android:screenOrientation="portrait"
            android:parentActivityName=".controller.anyadirAmigos.AnyadirAmigosActivity"
            android:theme="@style/NoStatusBar">
            <meta-data android:name="android.app.searchable" android:resource="@xml/searchable_config"/>
        </activity>
        <activity
            android:name=".controller.ranking.ApuestaFinalizada1x2Activity"
            android:label="MyBets"
            android:screenOrientation="portrait"
            android:theme="@style/NoStatusBar">
        </activity>
```

Figura 42: Extracto del AndroidManifest.xml de MyBets

```xml
<resources>

    <style name="MyActionBar.MenuTextStyle"
        parent="style/TextAppearance.AppCompat.Widget.ActionBar.Title">
    </style>

    <style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar"/>

    <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light"/>

    <style name="NoStatusBar" parent="Theme.AppCompat.NoActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:windowFullscreen">true</item>
        <item name="android:windowContentOverlay">@null</item>
    </style>

    <style name="MyEditTextTheme">
        <item name="colorControlNormal">#9999</item>
    </style>
```

Figura 43: Código de MyBets que muestra el formato de los estilos de la aplicación

### 5.1.3 Diseño

La guía de la API de Android [25] explica que los elementos de la interfaz de usuario se definen como objetos *View* y *ViewGroup*. Una *view* o vista es un objeto que dibuja algo en pantalla con lo que el usuario puede interactuar, mientras que un *ViewGroup* es un objeto vista que contiene otras vistas.

El diseño se puede realizar a través de instanciar objetos *view* en el código de las actividades y desarrollar árboles de vistas, sin embargo, es mucho más sencillo diseñar las pantallas utilizando archivos XML, como el ejemplo de la Figura 45.

Para poder usar los archivos de diseño, hay que cargarlos desde la actividad pertinente. Esto se realiza con una llamada a *setContentView()* donde se pasa la referencia al recurso (ver línea 71 de la Figura 46). Cuando se llama a una nueva actividad, el *framework* de Android llama al método *callback onCreate()* donde se realiza la llamada al recurso.

### 5.1.4 Uso de adaptadores para rellenar los datos de una vista

Para rellenar datos de ciertas vistas como por ejemplo las listas de elementos, se necesita enlazar la instancia de dicha lista en la Actividad a un *adapter*, que es el encargado de recuperar los datos. En el caso de MyBets, esos datos se recuperan de la base de datos Firebase. Para usar un *adapter* se pueden instanciar los que proporciona Android, sin embargo, debido a que en MyBets todos los elementos que se recogen de Firebase tienen un diseño personalizado, todas las clases *adapter* han sido programadas para cumplir con el diseño de los *mockups*. En la mayoría de casos se han creado los *adapter* como subclases de la clase *ArrayAdapter*. Para construir el *adapter* se pasan al

constructor como parámetros: el contexto de la aplicación, la referencia al recurso con el diseño que mostrará los datos, y la lista de datos a adaptar, como se puede ver en la línea 69 de la Figura 47.

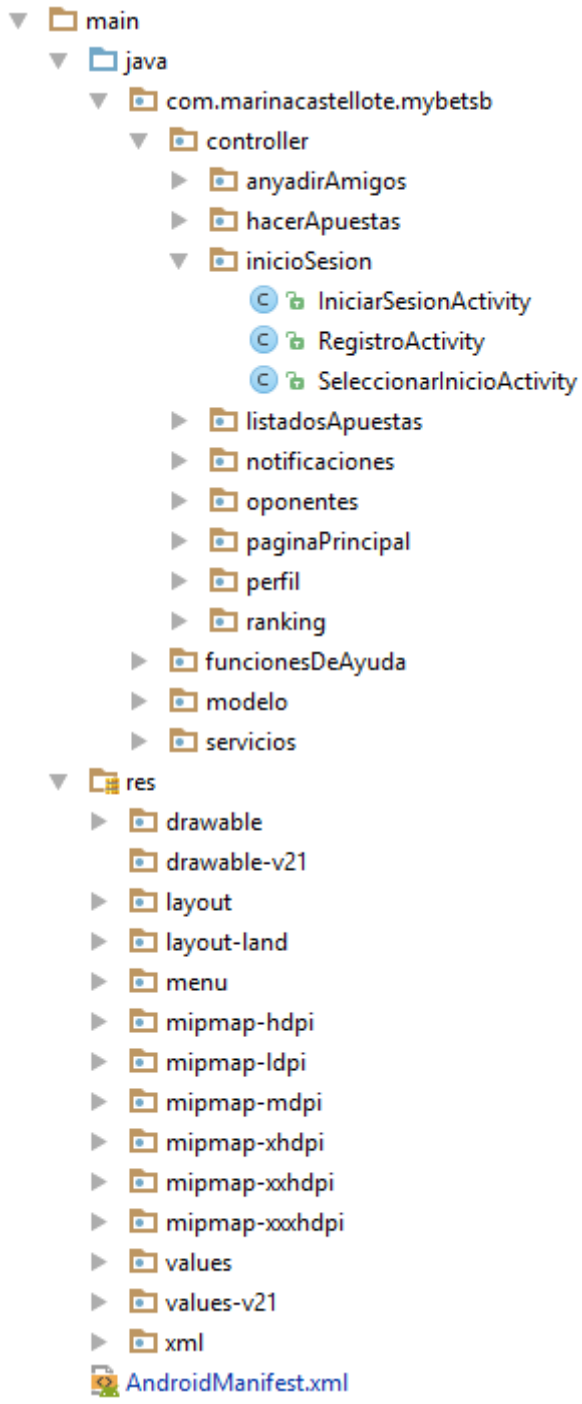


Figura 44: Extracto de la estructura del proyecto en Android Studio

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    android:orientation="vertical">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:paddingTop="15dp">

            <Button
                android:id="@+id/editarperfil_cerrar"
                android:layout_width="20dp"
                android:layout_height="20dp"
                android:layout_gravity="center_vertical"
                android:layout_marginLeft="15dp"
                android:layout_marginStart="15dp"
                android:background="@mipmap/cerrar_ed_perfil" />

```

Figura 45: Código de muestra del diseño de la vista de modificación de perfil

```java
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.editar_perfil);

        storage = FirebaseStorage.getInstance();

        displayName = (EditText) findViewById(R.id.ep_apodo);
        fullName = (EditText) findViewById(R.id.ep_nombre);
        email = (EditText) findViewById(R.id.ep_email);
        password = (EditText) findViewById(R.id.ep_password);
        image = (ImageView) findViewById(R.id.ep_imagen);
        boton = (Button) findViewById(R.id.ep_guardar);
        cerrar = (Button) findViewById(R.id.editarperfil_cerrar);
        cerrar.setOnClickListener(this);
        image.setOnClickListener(this);
        boton.setOnClickListener(this);
```

Figura 46: Código de MyBets que muestra cómo añadir una vista a una actividad

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.hoy);

    Bundle b = getIntent().getExtras();
    puntosUser = null;
    if (b != null) {
        if (b.getString("puntosUser") != null) {
            puntosUser = b.getString("puntosUser");
        }
    }

    mDatabase = FirebaseDatabase.getInstance().getReference();

    this.listView = (ListView) findViewById(R.id.listViewHoy);

    lista = new ArrayList<Categoria>();
    dataAdapter = new CategoriasAdapter(this, R.layout.categoria_item, lista, puntosUser);
    listView.setAdapter(dataAdapter);
```

Figura 47: Código de MyBets que muestra cómo añadir un *adapter* a una lista

### 5.1.5 Eventos

Como se explica en la guía de Android, los eventos son interfaces en la clase *View* que contienen métodos *callback*, los cuales son llamados por el *framework* de Android cuando una vista a la cual se le ha asignado un gestor de eventos es llamada debido a la interacción del usuario con el elemento. Por ejemplo, en la Figura 48 se muestra un extracto de código que ilustra este caso.

```

boton = (Button) findViewById(R.id.botonOkPorPuntos);
boton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int i = v.getId();
        if (i == R.id.botonOkPorPuntos) {
            if (dataAdapter.getSet().size() == lista.size()) {
                guardarDatos();
                DialogFragment newFragment = new EmApuestaEnviadaFragment();
                newFragment.show(getSupportFragmentManager(), "puntosEnviados");
            } else {
                Toast.makeText(getApplicationContext(), "Has de completar la apuesta",
                                                        Toast.LENGTH_SHORT).show();
            }
        }
    }
});
```

Figura 48: Código de MyBets que muestra el *callback* de un evento

### 5.1.6 Notificaciones

Una notificación es un mensaje que se muestra al usuario aunque la aplicación esté en segundo plano, apareciendo en el área de notificaciones del dispositivo.

Hay dos tipos de mensajes de notificaciones en Android, los mensajes *data* y los mensajes *notification*. Los de tipo *data* se pueden manejar en la aplicación tanto si la *app* está en primer como segundo plano. En cambio, los de tipo *notification* se tratan en la aplicación solo cuando está en primer plano. Si está en segundo plano, se genera una notificación automática, y cuando el usuario clica en la notificación, se le manda a la aplicación. Es por ello que al usar Cloud Functions de Firebase se envían a la aplicación mensajes de tipo data, sin embargo las notificaciones *push* que se hacen desde la interfaz de usuario Firebase son de tipo *notification* [26].

### 5.1.7 Lectura y escritura de la base de datos

Si se lee la documentación de Firebase, en concreto la parte que detalla la escritura [27] en la base de datos, se explica que para escribir datos en la base de datos de Firebase, la forma más básica se realiza mediante el uso del método *setValue()* a una referencia de la base de datos. De esta forma, los datos JSON serán modificados en el destino. Cuando se recibe un objeto personalizado de Java, solo es necesario que la clase del modelo tenga un constructor predeterminado sin argumentos. De esta forma, los atributos del objeto se asignarán a ubicaciones secundarias de la base de datos de forma anidada. En las líneas 65 y 66 de la Figura 49 se puede ver un ejemplo de uso de *setValue()*.

Para agregar datos a una lista de la base de datos se usa el método *push()*, que a la vez que agrega los datos genera un *id* único para que no haya conflictos de escritura.

```
56
57  private void guardarDatos() {
58
59      final HashMap<String, String> resul = dataAdapter.getSelected();
60      String modalidad = "1x2";
61
62      //creo mi apuesta y modifico partida publica
63      if (tipoPartida.equals("publica")) {
64          Integer p = Integer.parseInt(usuario.getPuntos()) - Integer.parseInt(puntos);
65          super.mDatabase.child("usuarios").child(FirebaseAuth.getInstance().getCurrentUser().getUid())
66                  .child("puntos").setValue(Integer.toString(p));
67          String key = super.mDatabase.child("apuestas").push().getKey();
            String keyResultado1x2 = super.mDatabase.child("resultados").push().getKey();
```

Figura 49: Código de MyBets que muestra la escritura en Firebase con setValue()

Por otro lado, la documentación también detalla [27] que para poder realizar la recuperación de datos se debe adjuntar un receptor asincrónico a una referencia de tipo *FirebaseDatabase*. De esta forma, el receptor se activará una vez para recoger el estado inicial de los datos y también cada vez que estos cambien en la base de datos de MyBets.

Hay diferentes tipos de receptores, cada uno de ellos con varios callbacks de evento:

- *ValueEventListener*: Se usa para detectar cambios en el contenido de una ruta de acceso.
- *ChildEventListener:* Se usa para detectar cambios en elementos de listas.

Si se desea que el receptor solo reciba el estado inicial, se agrega el receptor con el método *addListenerForSingleValueEvent(),* si se desea un uso continuo se usan los métodos *addValueEventListener()* o *addChildEventListener().* Los callbacks se retiran llamando a *removeEventListener().* En la figura 50 puede verse un ejemplo de uso de receptores.

```
30      mDatabase.child("apuestas").child(FirebaseAuth.getInstance()
31              .getCurrentUser().getUid()).addListenerForSingleValueEvent(new ValueEventListener() {
32          @Override
33          public void onDataChange(DataSnapshot dataSnapshot) {
34              GenericTypeIndicator<HashMap<String, Apuesta>> mapType =
35                      new GenericTypeIndicator<HashMap<String, Apuesta>>() { };
36              HashMap<String, Apuesta> map = dataSnapshot.getValue(mapType);
37              Apuesta apuesta = null;
38              if (map != null) {
39                  for (Apuesta ap : map.values()) {
40                      if (ap.getIdPartida().equals(partida.getId())) {
41                          apuesta = ap;
42                      }
43                  }
44              }
```

Figura 50: Código de MyBets que muestra el uso de receptores en la lectura de datos de Firebase

### 5.1.8 Autenticación

Para reconocer la identidad del usuario, Firebase proporciona Firebase Authentication, con servicios SDK y bibliotecas preparadas para autenticar usuarios con diferentes proveedores de identidad. En Mybets se ha usado la autenticación basada en correo electrónico y contraseña, así como el proveedor de identidad de Facebook.

Cuando un usuario inicia sesión con unas credenciales, estas se pasan al SDK de Firebase Authentication. El *backend* de Firebase las comprueba y devuelve la respuesta al cliente.

### 5.1.9 Compatibilidad

Android funciona de forma que aplicaciones antiguas siempre serán compatibles con nuevas versiones de Android. Además, MyBets ha sido desarrollado para una versión mínima de SDK de 16, lo que significa que todos los dispositivos Android que usen un nivel de API 16 o superior (es decir, una versión Android 4.1 o superior) serán siempre compatibles con la aplicación.

## 5.2 Verificación y validación

La verificación y validación de la aplicación se ha realizado en cada una de las revisiones de los sprints.

Se han realizado tanto pruebas estáticas, revisando el código desarrollado durante el sprint línea a línea, como dinámicas, ejecutando la aplicación. Dentro de las pruebas dinámicas se han realizado pruebas de caja negra, comprobando que la aplicación cumplía tanto los requisitos funcionales como los no funcionales, asegurando que la interfaz de usuario cumple los principios de Material Design y no provoca confusión al usuario, sino que este puede manejar la aplicación de forma intuitiva. En los pocos casos en los que el usuario introduce valores directamente, se ha comprobado que todas las opciones de validación de introducción de datos programadas se ejecutan en el caso correspondiente. Por ejemplo, cuando el usuario trata de apostar un valor de puntos mayor de los que dispone o menor de 50, no se le permite.

Las pruebas funcionales y no funcionales han sido realizadas de forma manual, con un dispositivo físico Samsung Galaxy S6, el cual utiliza la versión de Android 6.0 Marshmallow. También se han hecho pruebas con el emulador de Android Studio para probar resoluciones mayores.

Las pruebas manuales se han hecho como sigue:

1. Se ha comprobado específicamente el cumplimiento de las historias de usuario planteadas en la pila del producto. Esto ha permitido comprobar la validación de las funcionalidades, asegurando que la aplicación cumple los requisitos establecidos.

2. A continuación se ha probado de forma general el funcionamiento, verificando que se obtienen los resultados correctos en cada prueba:

    2.1 En primer lugar se ha comprobado el redireccionamiento de las pantallas, así como la actualización de datos en estas.

    2.2 Se ha comprobado que las listas de datos se pueblan correctamente según los datos que existen en la base de datos.

    2.3 La última comprobación ha sido ver que los eventos disparados debido a acciones del usuario provocaban los efectos esperados tanto en la interfaz, como en la escritura de la base de datos.

3. Si alguno de estos procesos ha fallado, se ha anotado como incidencia para el siguiente sprint.

# Capítulo 6

# Conclusiones

En primer lugar quiero decir que considero cumplidos los objetivos que me había planteado para el proyecto, ya que además de haber extendido mis conocimientos de programación, he aumentado mis habilidades de análisis y toma de decisiones, siendo esta la primera vez que me encargo del desarrollo completo de una aplicación de este tamaño. Aunque nunca antes había programado para el sistema operativo Android ni utilizado la plataforma Firebase, gracias a los conocimientos previos de Java que he desarrollado durante el estudio del grado, se me ha facilitado la tarea de desarrollo.

También cabe destacar que parte del éxito al cumplir las estimaciones de duración y complejidad del proyecto han sido gracias al uso de metodología ágil Scrum, que me ha ayudado a organizar el proyecto teniendo siempre en cuenta la carga de trabajo restante y qué partes del desarrollo debía priorizar.

Respecto al uso de Firebase, me ha parecido una plataforma muy útil para aplicaciones como MyBets, las cuales no necesitan una gran cantidad de código *backend*, y espero tener la oportunidad de volver a usarla en el futuro en proyectos profesionales.

Otra de las oportunidades más interesantes del desarrollo de este trabajo es el haber tenido la oportunidad de conocer cómo trabaja una empresa dedicada a la programación en su día a día, cómo llevan a cabo la gestión de sus proyectos y la colaboración entre los equipos de trabajo.

Respecto a las dificultades encontradas a la hora de desarrollar el proyecto, he de destacar como partes más complejas del proceso el tener que usar una base de datos noSQL, ya que hasta el momento me había desenvuelto sobre todo con bases de datos relacionales, lo cual ha supuesto un aumento en la complejidad a la hora de estructurar los datos y las peticiones a la base de datos. Sin embargo, a medida que avanzaba el periodo de desarrollo he podido ver como mis competencias mejoraban.

Como posibles extensiones de esta aplicación incluiría el uso de una API de apuestas que automatice la inserción de los resultados finales en la base de datos de Firebase. Otra posibilidad es el paso de este modelo de negocio como aplicación de entretenimiento a una aplicación de pago segura.

En cuanto a la viabilidad comercial de la aplicación, pienso que MyBets podría tener futuro como alternativa de ocio gratuito a las casas de apuestas reales, mediante la inserción de anuncios publicitarios, para aquellas personas que disfrutan de los juegos de azar pero prefieren no arriesgar dinero real.

El auge actual de las aplicaciones móviles ha aumentado la demanda de programadores de *apps* considerablemente. Considero que el desarrollo de este proyecto ha permitido aumentar no solo mis conocimientos de programación, sino también la posible orientación de un camino profesional, por lo menos para el futuro próximo, en el que planeo seguir programando aplicaciones Android a través de un contrato de prácticas extracurriculares en la empresa Cuatroochenta. Sin duda recomendaría a otros estudiantes la elección de esta empresa para la estancia en prácticas, ya que por lo que he podido observar durante mi estancia, sus trabajadores hacen siempre lo posible por ayudar en la formación de los alumnos que pasan por ella, demostrando consideración en la flexibilidad de horarios y un gran ambiente de trabajo.

# Bibliografía


[1] Soluciones Cuatroochenta S.L. *Sefici.*

Consultada el 10 de mayo de 2017, en http://www.cuatroochenta.com/portfolio/sefici/

[2] Soluciones Cuatroochenta S.L. *480interactive.*

Consultada el 10 de mayo de 2017, en http://www.cuatroochenta.com/portfolio/480interactive/

[3] Hackaton Castellón (2017).

Consultada el 10 de mayo de 2017, en http://www.hackathoncastellon.com/

[4] David Serrano Hernández (2016). *Realización de los servicios web para una aplicación móvil de apuestas deportivas*. TFG del Grado de Informática, curso 2015/2016.

Consultada el 28 de junio de 2017, en http://repositori.uji.es/xmlui/handle/10234/166715

[5] Ignacio Vicent Salvador (2016). *Desarrollo de aplicación iOS utilizando webservices REST en formato JSON*. TFG del Grado de Informática, curso 2015/2016.

Consultada el 28 de junio de 2017, en http://repositori.uji.es/xmlui/handle/10234/166708

[6] Catalín Denís Damián (2016). *Desarrollo de aplicación Android para proyecto de apuestas deportivas*. TFG del Grado de Informática, curso 2015/2016.

Consultada el 28 de junio de 2017, en http://repositori.uji.es/xmlui/handle/10234/167043

[7] Documentación Firebase (2016). *Firebase Authentication*.

Consultada el 28 de junio de 2017, en https://firebase.google.com/docs/auth/

[8] Documentación Firebase (2017). *Understand Firebase Realtime Database Rules*.

Consultada el 28 de junio de 2017, en https://firebase.google.com/docs/database/security/

[9] Documentación Android. *Material Design for Developers.*

Consultada el 5 de junio de 2017, en https://developer.android.com/training/material/index.html

[10] Documentación Android Studio (2017). *Android Studio. The Official IDE for Android.*

Consultada el 5 de junio de 2017, en https://developer.android.com/studio/index.html?#features

[11] Documentación Firebase. *Firebase helps you build better mobile apps and grow your business.*

Consultada el 14 de junio de 2017, en https://firebase.google.com/

[12] Scrum.org (2017). *What is Scrum?*

Consultada el 7 de junio de 2017, en https://www.scrum.org/resources/what-is-scrum

[13] Xavier Quesada Allue. *Introducción a la estimación y planificación ágil.*

Consultada el 8 de junio de 2017, en https://proyectosagiles.org/2009/06/08/introduccion-estimacion-planificacion-agil/

[14] Atlassian (2017). *Jira Software*.

Consultada el 10 de junio de 2017, en https://es.atlassian.com/software/jira

[15] Atlassian (2017). *Bitbucket*.

Consultada el 10 de junio de 2017, en https://bitbucket.org/

[16] Adobe (2017). *Adobe Illustrator CC.*

Consultada el 10 de junio de 2017, en https://www.adobe.com/es/products/illustrator.html

[17] Udemy (2017). *Udemy*.

Consultada el 10 de junio de 2017, en https://www.udemy.com/

[18] Firebase. *Pricing plans.*

Consultada 10 de junio de 2017, en https://firebase.google.com/pricing/

[19] CincoDías (2017). *Euros x Dólares USA.*

Consultada el 28 de junio de 2017, en http://cincodias.com/mercados/divisas/eurosxdolares_usa/41/

[20] Documentación Firebase (2017). Add *Firebase to your Android Project.*

Consultada el 10 de junio de 2017, en https://firebase.google.com/docs/android/setup

[21] Documentación Google Cloud Platform (2017). *Mobile App Backend Services.*

Consultada el 14 de junio de 2017, en https://cloud.google.com/solutions/mobile/mobile-app-backend-services

[22] Firebase. *Firebase Realtime Database. Store and sync data in real time.*

Consultada el 14 de junio de 2017, en https://firebase.google.com/products/database/

[23] Documentación Firebase (2017). *Structure Your Database.*

Consultada el 15 de junio de 2017, en https://firebase.google.com/docs/database/web/structure-data?

[24] Guías Android. *Application Fundamentals.*

Consultada el 15 de junio de 2017, en https://developer.android.com/guide/components/fundamentals.html

[25] Documentación Android. *Layouts.*

Consultada el 15 de junio de 2017, en https://developer.android.com/guide/topics/ui/declaring-layout.html

[26] Documentación Firebase (2017). *About FCM Messages.*

Consultada el 15 de junio de 2017, en https://firebase.google.com/docs/cloud-messaging/concept-options

[27] Documentación Firebase (2017). *Read and Write Data on Android.*

Consultada el 15 de junio de 2017, en https://firebase.google.com/docs/database/android/read-and-write

# Anexo A

# Diagrama de Gantt

En las Figuras 51, 52 y 53 se muestra el diagrama de Gantt realizado en la planificación inicial del proyecto de forma desglosada, mientras que en la Figura 54 se puede ver en su totalidad.

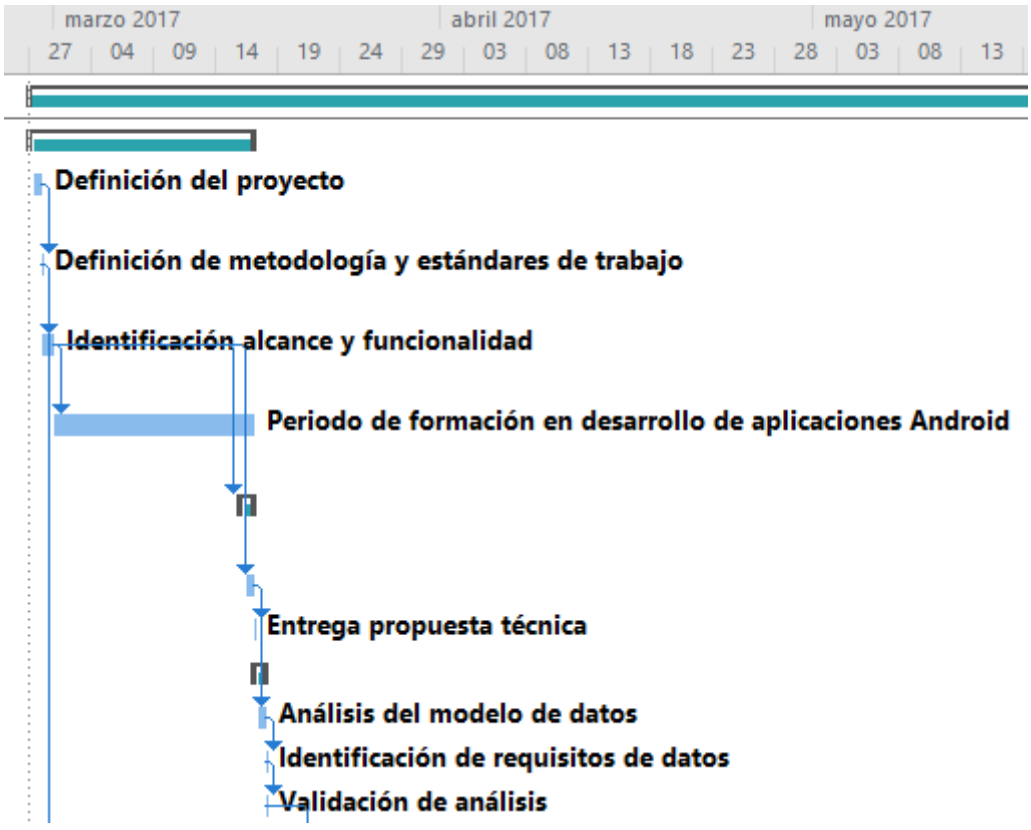


Figura 51: Diagrama de Gantt desglosado (I)

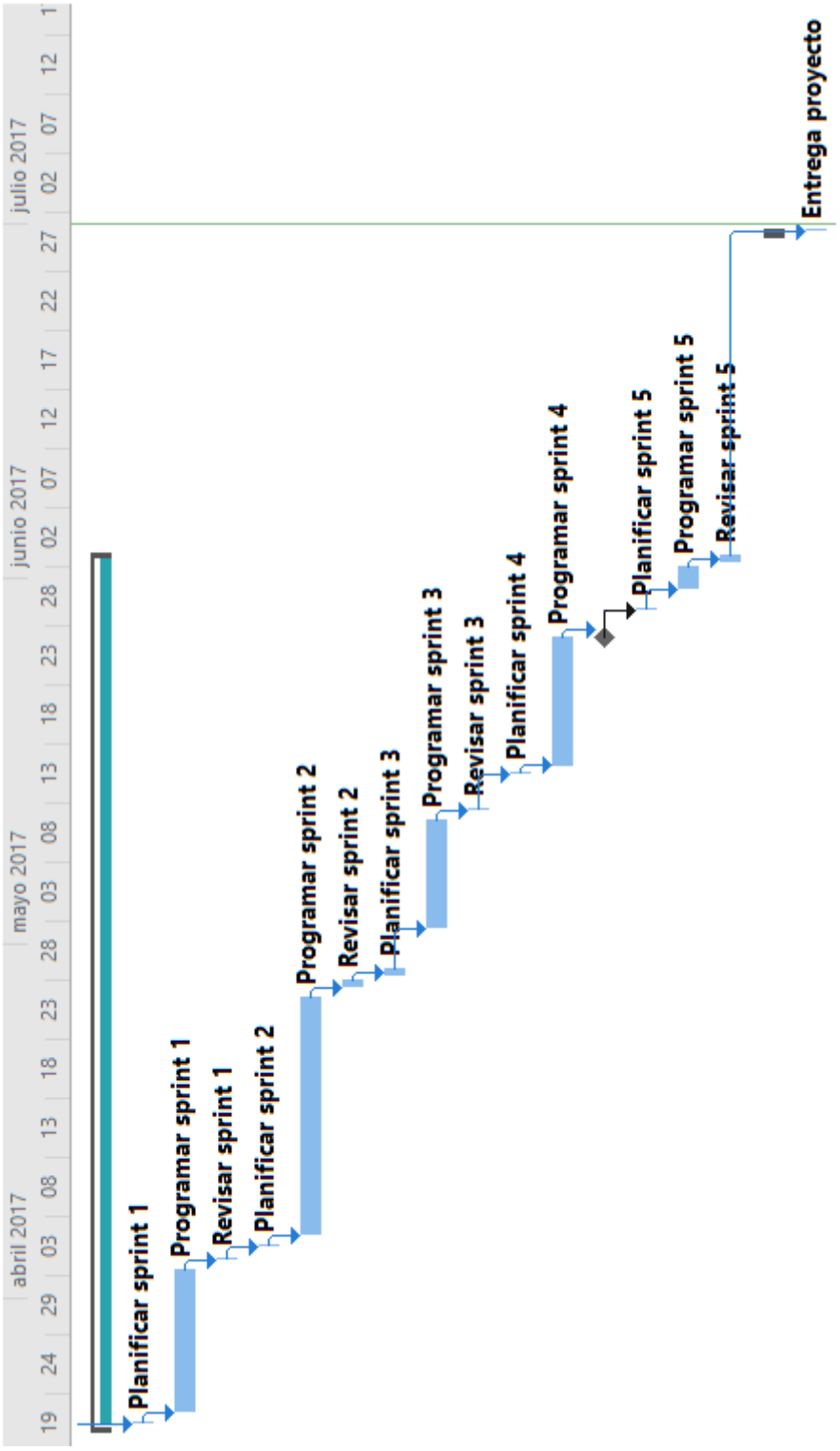


Figura 52: Diagrama de Gantt desglosado (II)

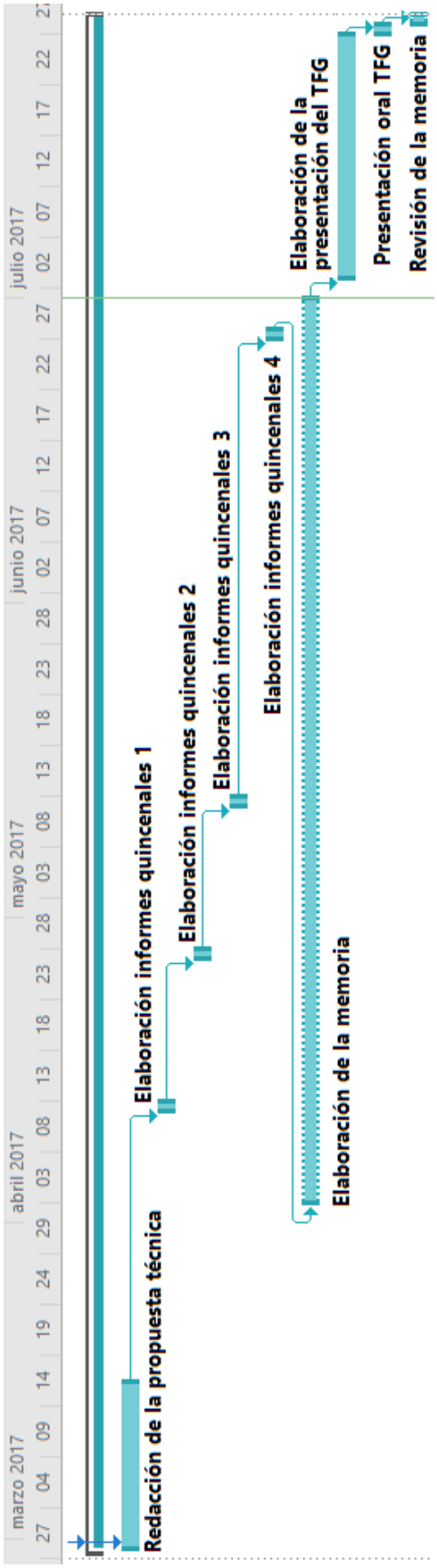


Figura 53: Diagrama de Gantt desglosado (III)

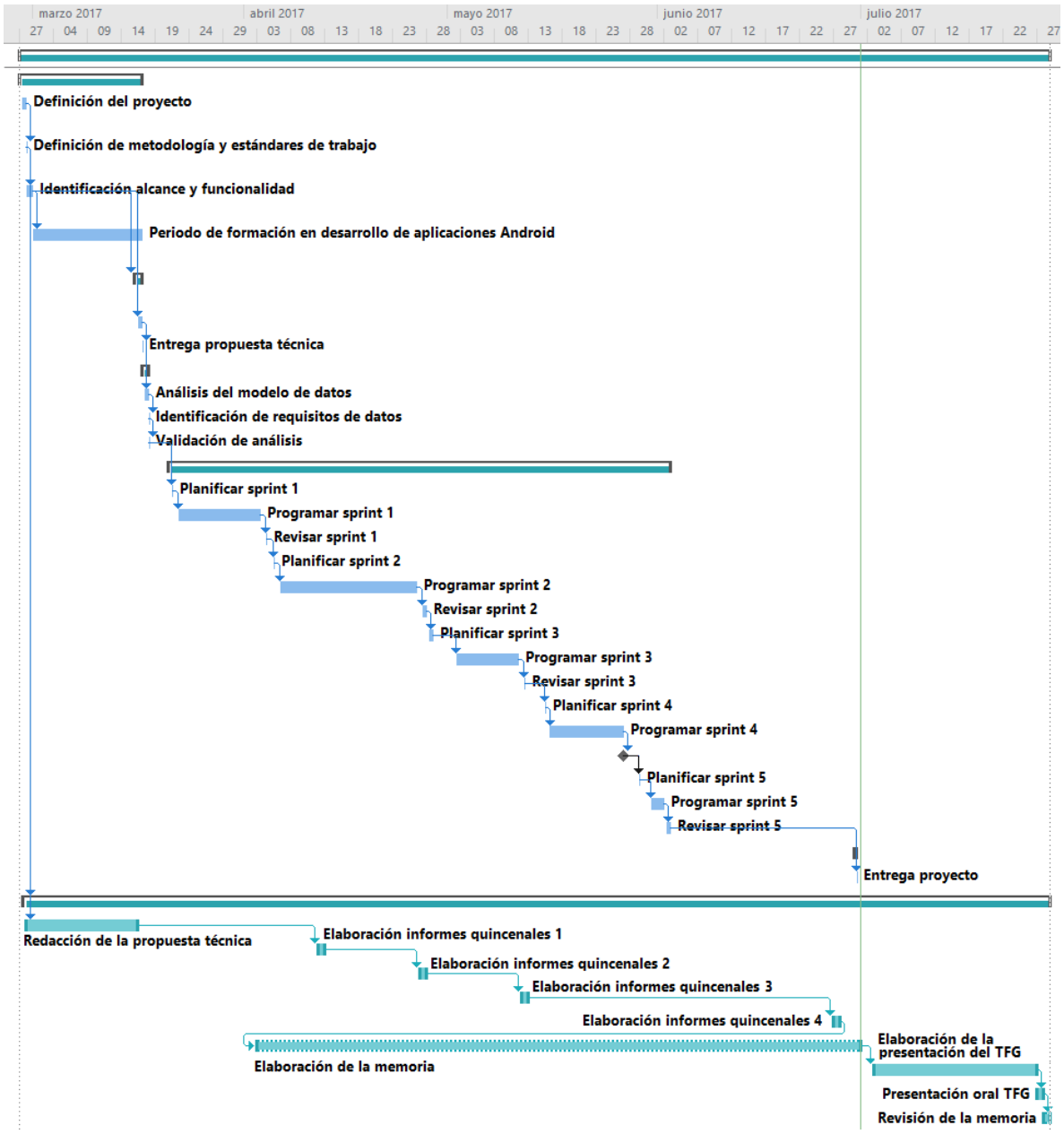


Figura 54: Diagrama de Gantt completo

# Anexo B

# Esquema de la base de datos

En las Figuras 55 a 61 de este anexo se puede observar el esquema de la base de datos noSQL tal y como se muestra en la interfaz de usuario de la web de Firebase. Cabe señalar que estas figuras son solo un resumen de la base de datos completa.

En la Figura 55 se pueden ver los nodos *actividadPerfil* y *amigos:*

- *actividadPerfil: G*uarda los mensajes que se muestran en la actividad del perfil de usuario, organizados por *id* de usuario. Los datos guardados son la fecha en la que se produjo la actividad, el texto del mensaje y los puntos ganados o perdidos.
- *amigos*: Guarda los identificadores de los usuarios que han sido agregados a la lista de amigos de otro usuario.

En la Figura 56 se muestran los nodos *apuestas*, *competicionesSeguidas*, *enfrentamientos* y *categorias*.

- *apuestas*: Almacena los datos de las apuestas que han sido realizadas, incluyendo información sobre si la apuesta ya ha sido hecha (cuando *estadoJugada* es *false* significa que es una apuesta creada por invitación de otro usuario), si se ha aceptado o rechazado en el caso de ser una invitación, los identificadores de la partida y el usuario vinculados a la apuesta, el identificador que apunta a los resultados seleccionados, la modalidad de la apuesta, los puntos apostados por el usuario, y si es pública o privada.
- *competicionesSeguidas*: Se guardan los *tokens* de los dispositivos a los que se informará cuando se tengan los resultados de la partida.
- *enfrentamientos*: Guarda las listas de datos de enfrentamientos de diferentes competiciones deportivas. Los datos *item1* e *item2* hacen referencia a los ítems o contrincantes que juegan ese enfrentamiento. El dato *idItems* guarda el identificador de la lista donde se encuentran los datos de dichos ítems.
- *categorias***:** Guarda los datos de las diferentes categorías de apuestas. Esto incluye las url con las imágenes de fondo e iconos que representan dicha categoría, así como el nombre.

En la Figura 57 se observan los siguientes nodos:

- *items*: Listas con las posibles elecciones de resultados de cada partida.
- *muro***:** Listado de la actividad de los amigos de un usuario. Incluye la misma información que *actividadPerfil*, y además el identificador del usuario que ha realizado la actividad.

En las Figuras 58 y 59 se muestran los nodos *partidasPrivadas* y *partidasPublicas* respectivamente.

- *partidasPrivadas*: Datos de las partidas privadas creadas por los usuarios. Guarda el bote de la partida, la categoría, si se tienen ya los resultados de la partida o no, la fecha de fin de partida, la referencia a la lista de ítems o contrincantes de la partida, la lista con los identificadores de las apuestas realizadas, la lista con los identificadores de los usuarios invitados a la partida, la lista de usuarios que ya han participado en la partida, la modalidad, el nombre, la fecha de publicación y la subcategoría a la que pertenece.
- *partidasPublicas*: Datos de las partidas públicas de la aplicación. Guarda los mismos datos que *partidaPrivada* menos la lista y número de invitados, que no tiene.

La Figura 60 muestra los nodos *resultados*, *resultadosCorrectos* y *subcategorias*.

- *resultados*: Guarda los resultados elegidos de cada apuesta. Cada vez que un usuario realiza una apuesta, aquí se guarda su selección de resultados.
- *resultadosCorrectos*: En este nodo se almacenan los resultados finales correctos de las partidas, los cuales se usan para saber los puntos que debe ganar cada participante.
- *subcategorias*: Datos de las subcategorías de apuestas organizados según la categoría a la que pertenecen.

Finalmente, la Figura 61 muestra los nodos *tokens* y *usuarios*.

- *tokens*: Lista de identificadores de los dispositivos de cada usuario. Se guardan según el usuario al que pertenecen los dispositivos.
- *usuarios*: Datos de los usuarios de la aplicación. Se guarda el nombre completo, el nombre de usuario, el identificador de la cuenta de Facebook vinculada si la hay, la foto de perfil y los puntos acumulados. El correo y la contraseña son guardados a parte por el servicio Firebase Authentication, y están vinculados al identificador de usuario.

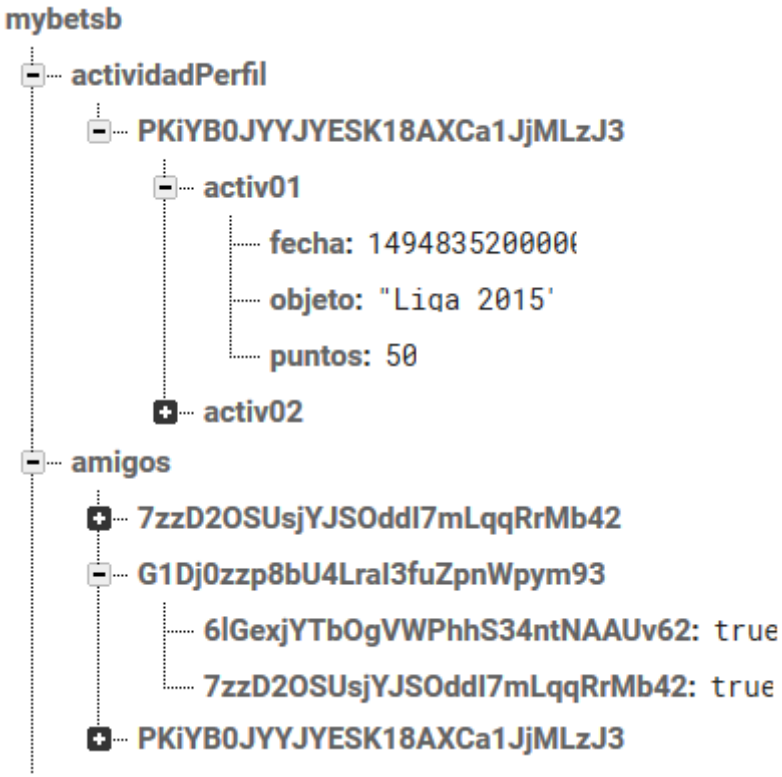


Figura 55: Resumen de la estructura de la base de datos de MyBets (I)


Figura 56: Resumen de la estructura de la base de datos de MyBets (II)

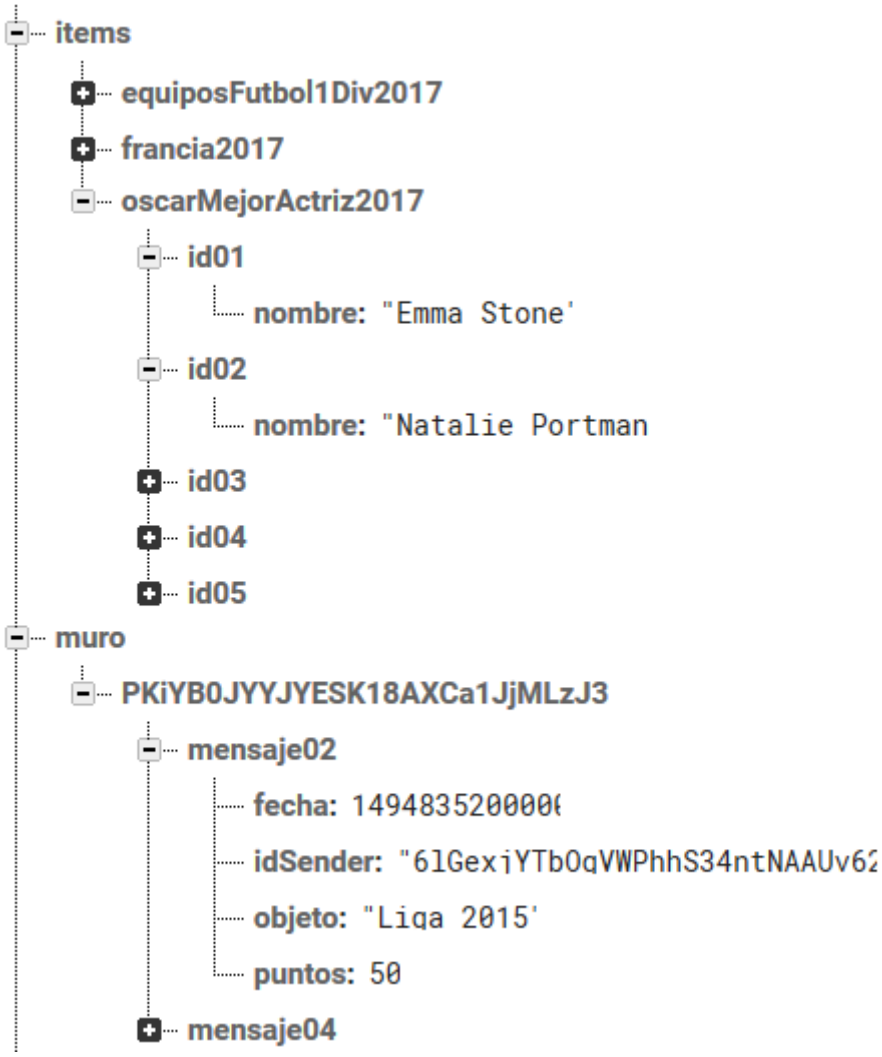


Figura 57: Resumen de la estructura de la base de datos de MyBets (III)


Figura 58: Resumen de la estructura de la base de datos de MyBets (IV)


Figura 59: Resumen de la estructura de la base de datos de MyBets (V)

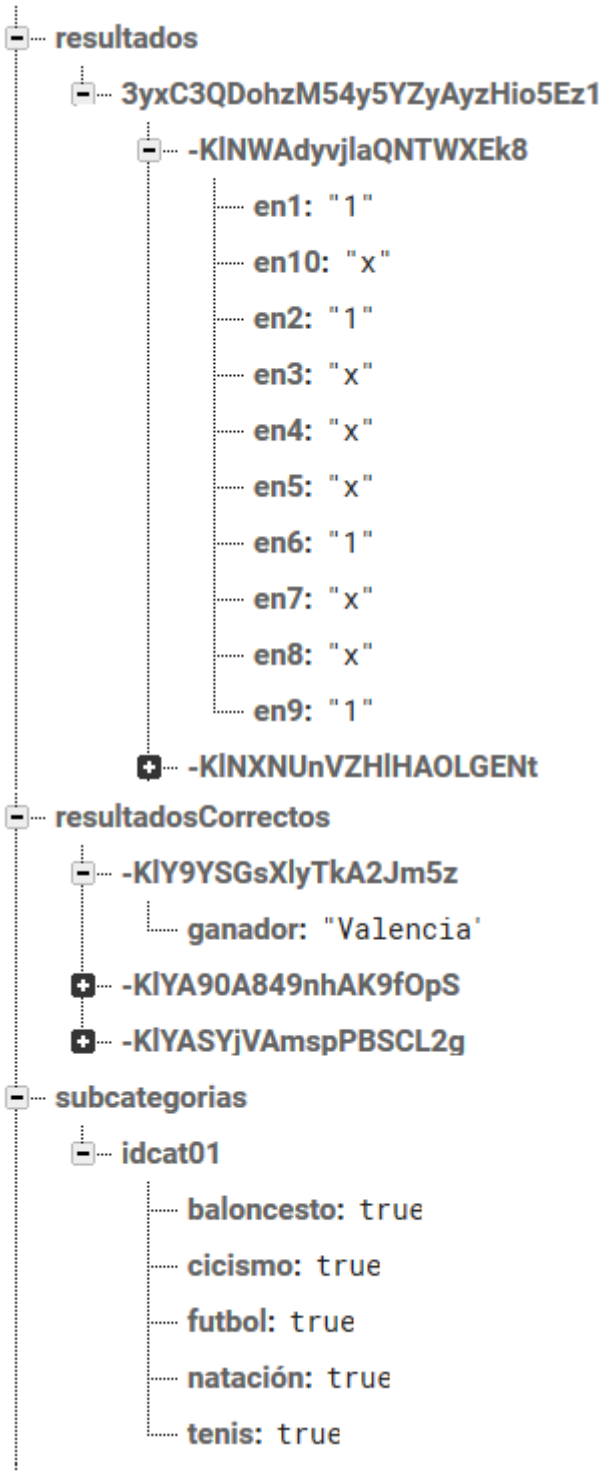


Figura 60: Resumen de la estructura de la base de datos de MyBets (VI)

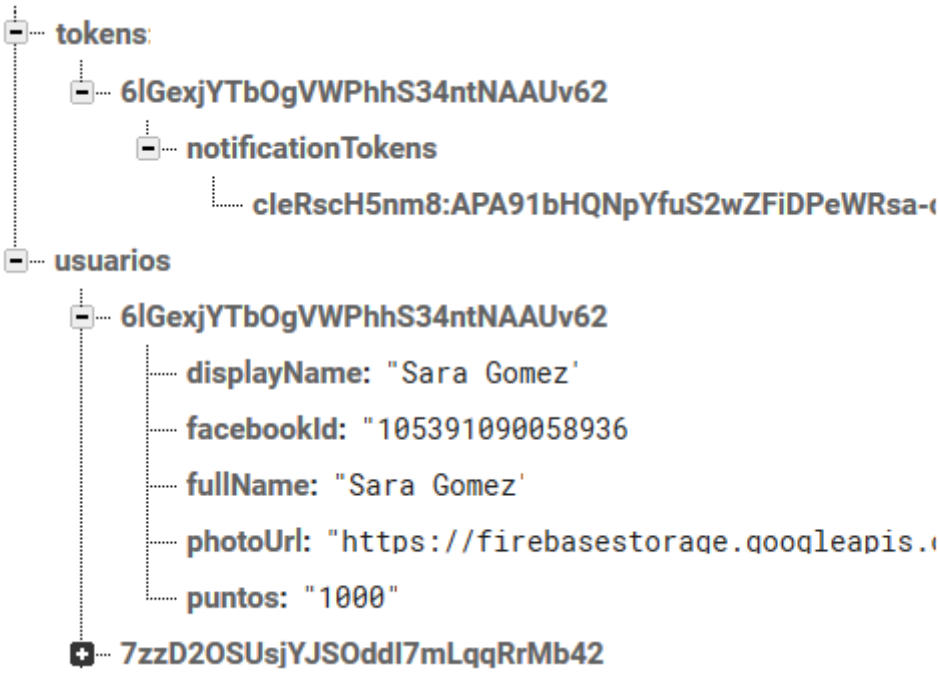


Figura 61: Resumen de la estructura de la base de datos de MyBets (VII)

# Anexo C

# Lista de mockups proporcionados por la empresa

A continuación se listan todos los *mockups* proporcionados a la alumna, realizados por el equipo de diseño de Cuatroochenta, representados en las Figuras 62 a 90.


Figura 62: *Mockup* 1 MyBets

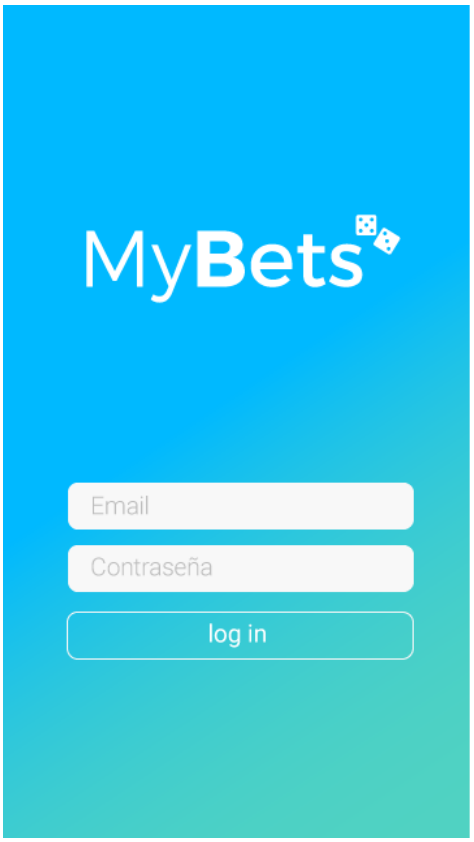


Figura 63: *Mockup* 2 MyBets

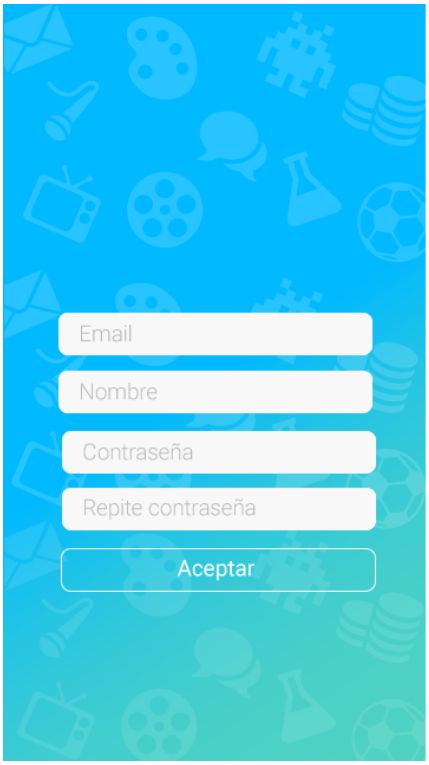


Figura 64: *Mockup* 3 MyBets

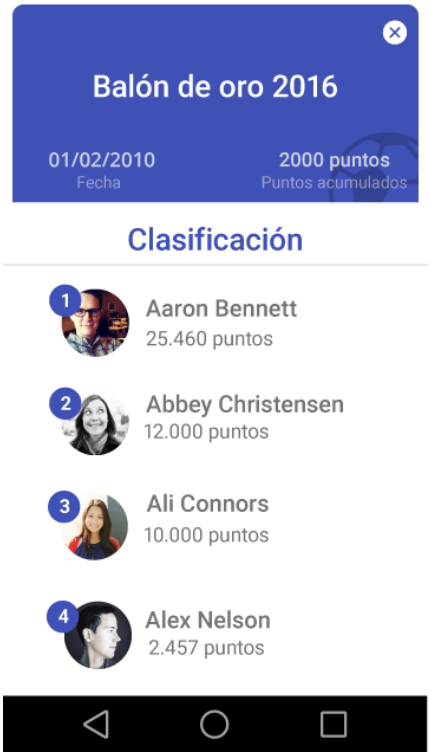


Figura 65: *Mockup* 4 MyBets

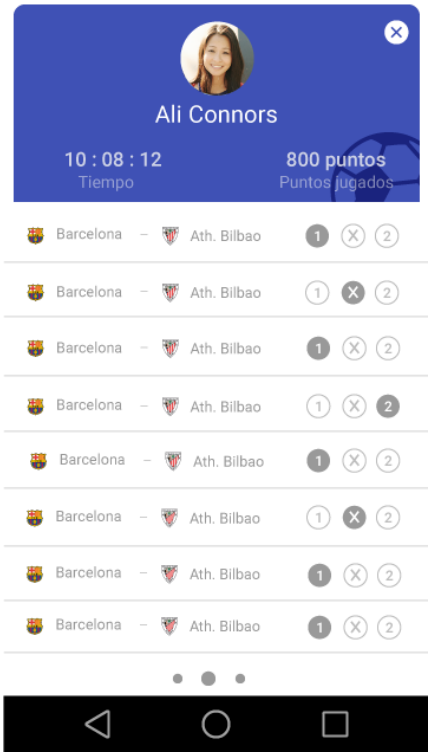


Figura 66: *Mockup* 5 MyBets

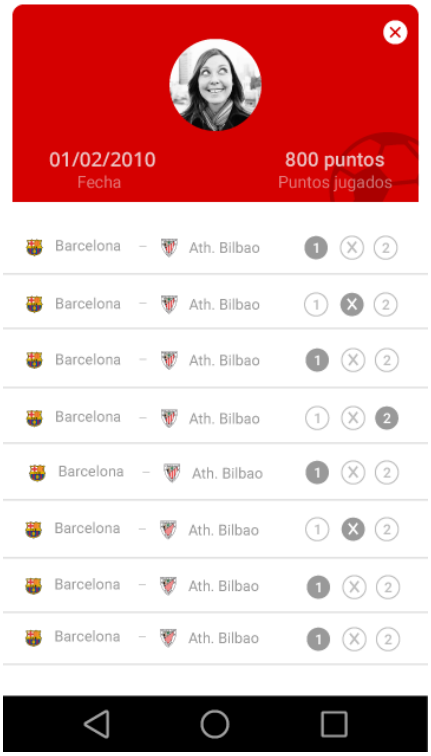


Figura 67: *Mockup* 6

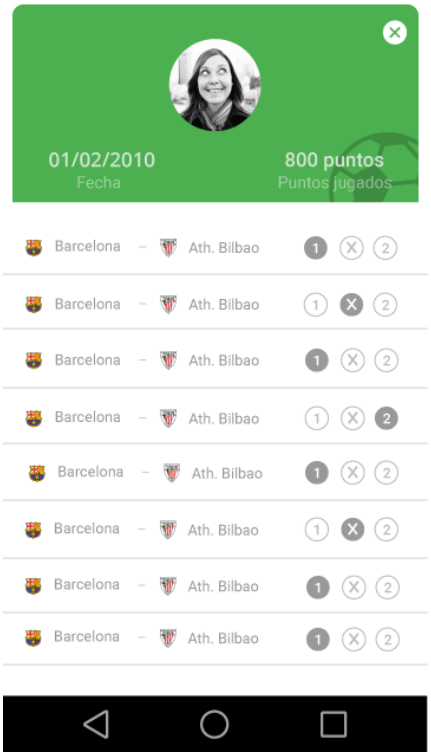


Figura 68: *Mockup* 7 MyBets

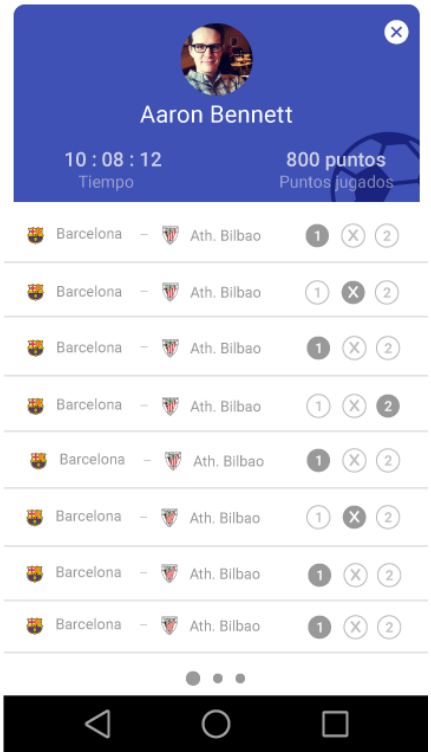


Figura 69: *Mockup* 8 MyBets

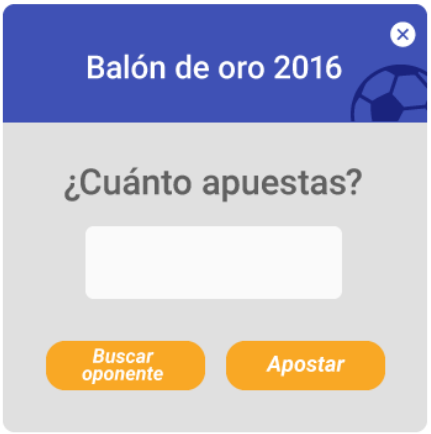


Figura 70: *Mockup* 9 MyBets

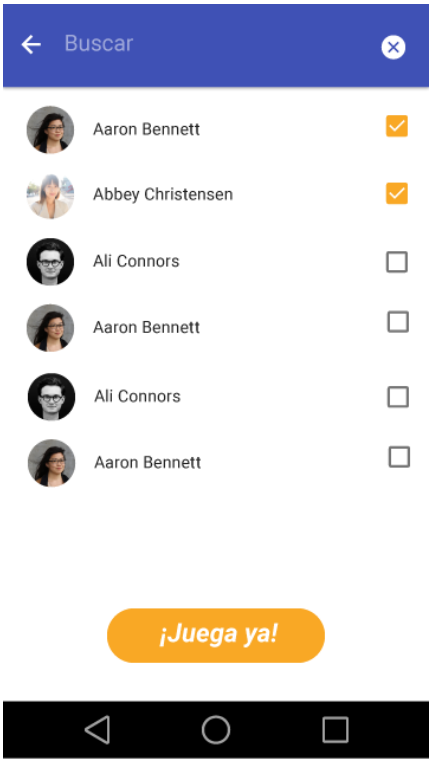


Figura 71: *Mockup* 10 MyBets

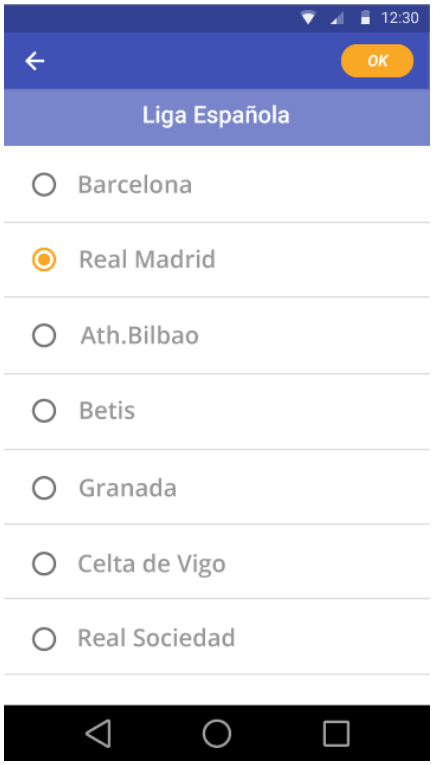


Figura 72: *Mockup* 11 MyBets

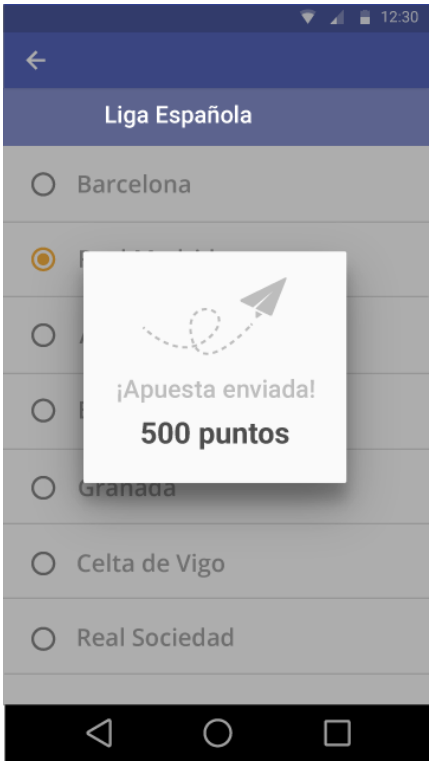


Figura 73: *Mockup* 12 MyBets


Figura 74: *Mockup* 13 MyBets


Figura 75: *Mockup* 14 MyBets

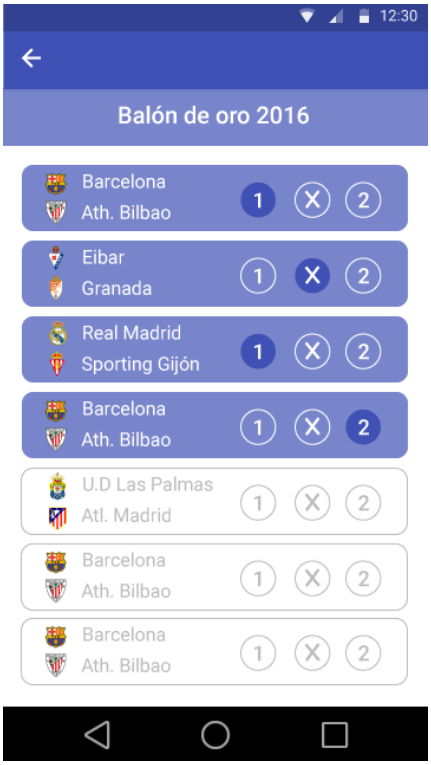


Figura 76: *Mockup* 15 MyBets


Figura 77: *Mockup* 16 MyBets

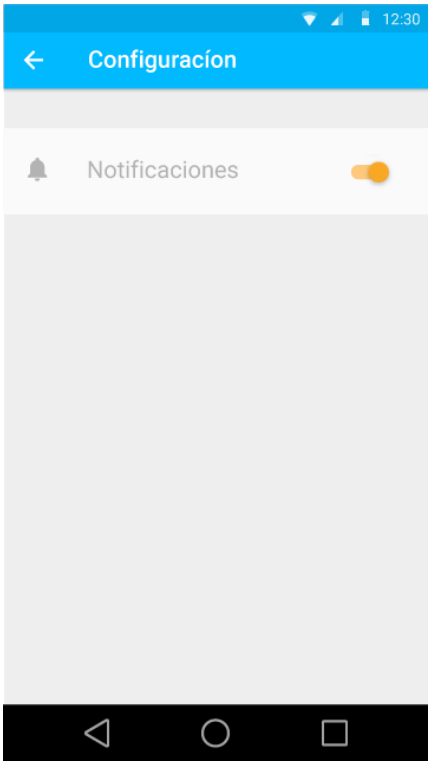


Figura 78: *Mockup* 17 MyBets


Figura 79: *Mockup* 18 MyBets


Figura 80: *Mockup* 19 MyBets

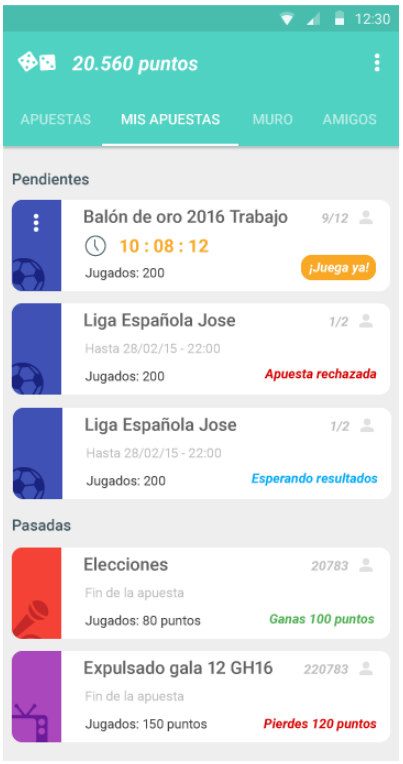


Figura 81: *Mockup* 20 MyBets

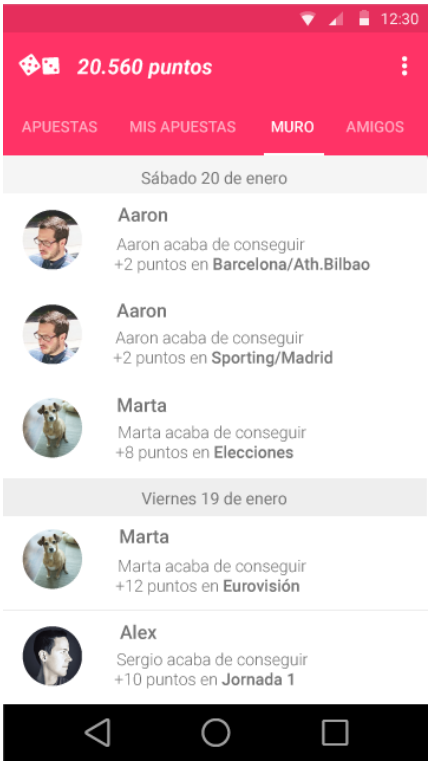


Figura 82: *Mockup* 21 MyBets

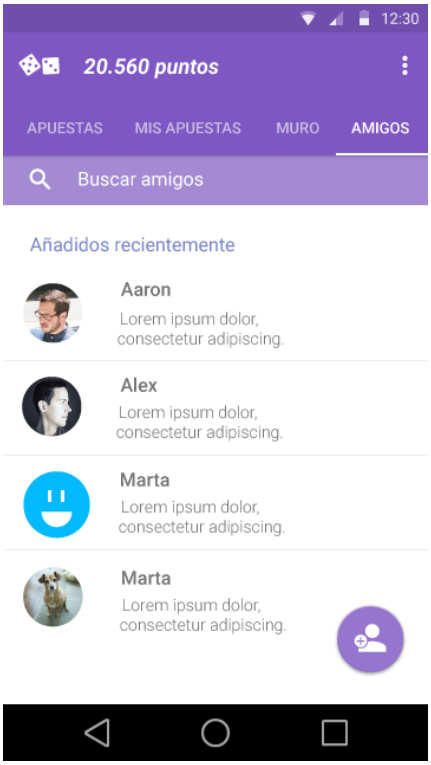


Figura 83: *Mockup* 22 MyBets


Figura 84: *Mockup* 23 MyBets


Figura 85: *Mockup* 24 MyBets

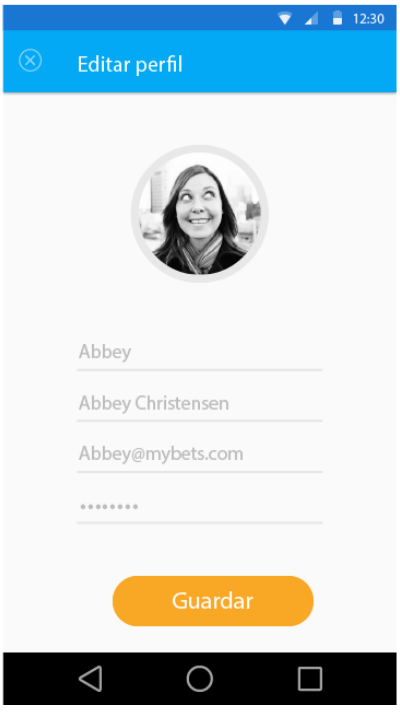


Figura 86: *Mockup* 25 MyBets

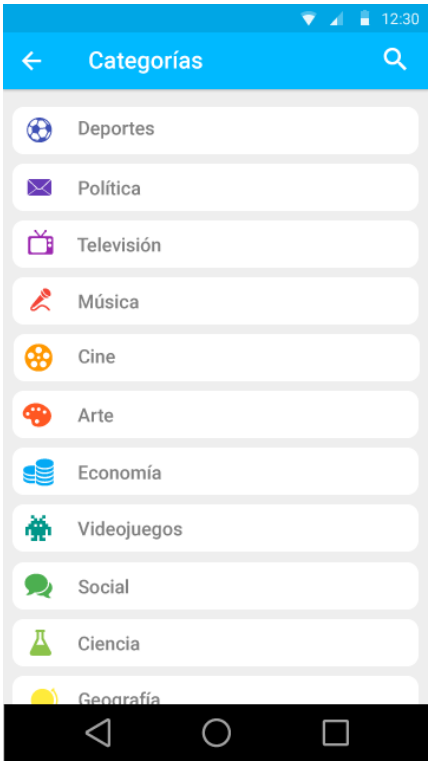


Figura 87: *Mockup* 26 MyBets

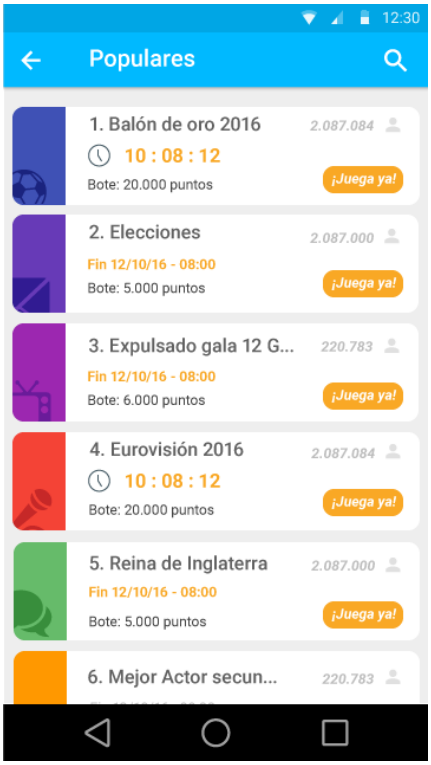


Figura 88: *Mockup* 27 MyBets

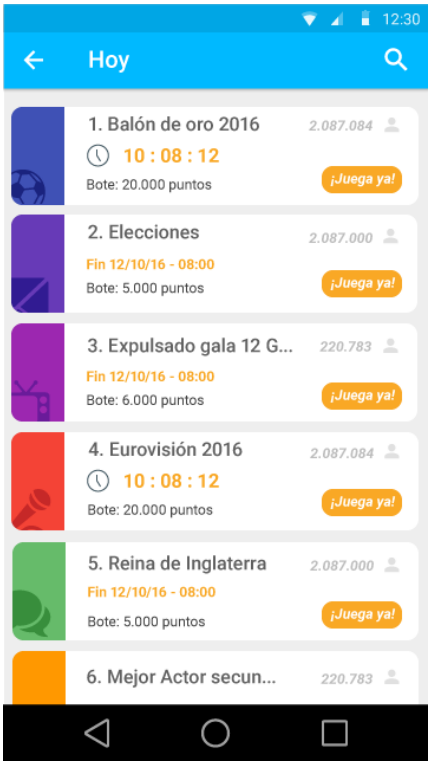


Figura 89: *Mockup* 28 MyBets

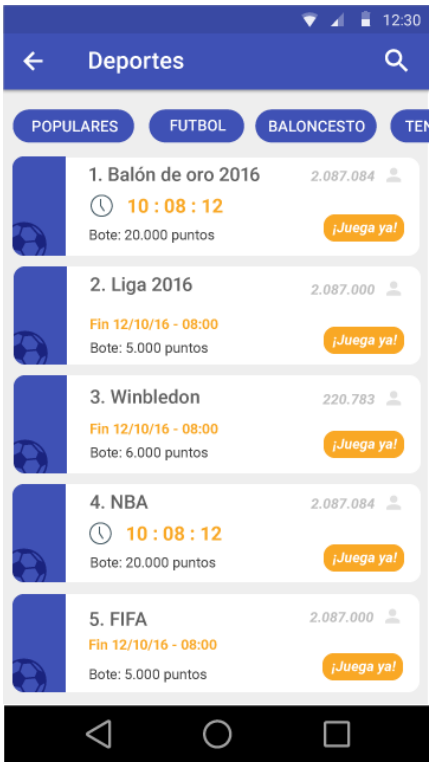


Figura 90: *Mockup* 29 MyBets